

SQL PRACTICE -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.

1. Retrieve all columns from the product table.

QUERY:

```
create database ganesh;
```

```
use ganesh;
```

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
```

```
(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),
```

```
(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

QUERY WITH OUTPUT:

The screenshot displays a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 create database ganesh;
2 use ganesh;
3 CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 select *from Products;
5 INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
6 (101, 'Laptop', 'Electronics', 500.00),
7 (102, 'Smartphone', 'Electronics', 300.00),
8 (103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
9 (105, 'Mouse', 'Electronics', 15.00);
10
```

The result grid shows the output of the query, displaying the product details:

product_id	product_name	category	unit_price
101	Laptop	Electronics	500.00
102	Smartphone	Electronics	300.00
103	Headphones	Electronics	30.00
104	Keyboard	Electronics	20.00
105	Mouse	Electronics	15.00
NULL	NULL	NULL	NULL

2. Retrieve the product_name and unit_price from the Products table.

QUERY:

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
```

```
(101, 'Laptop', 'Electronics', 500.00),
```

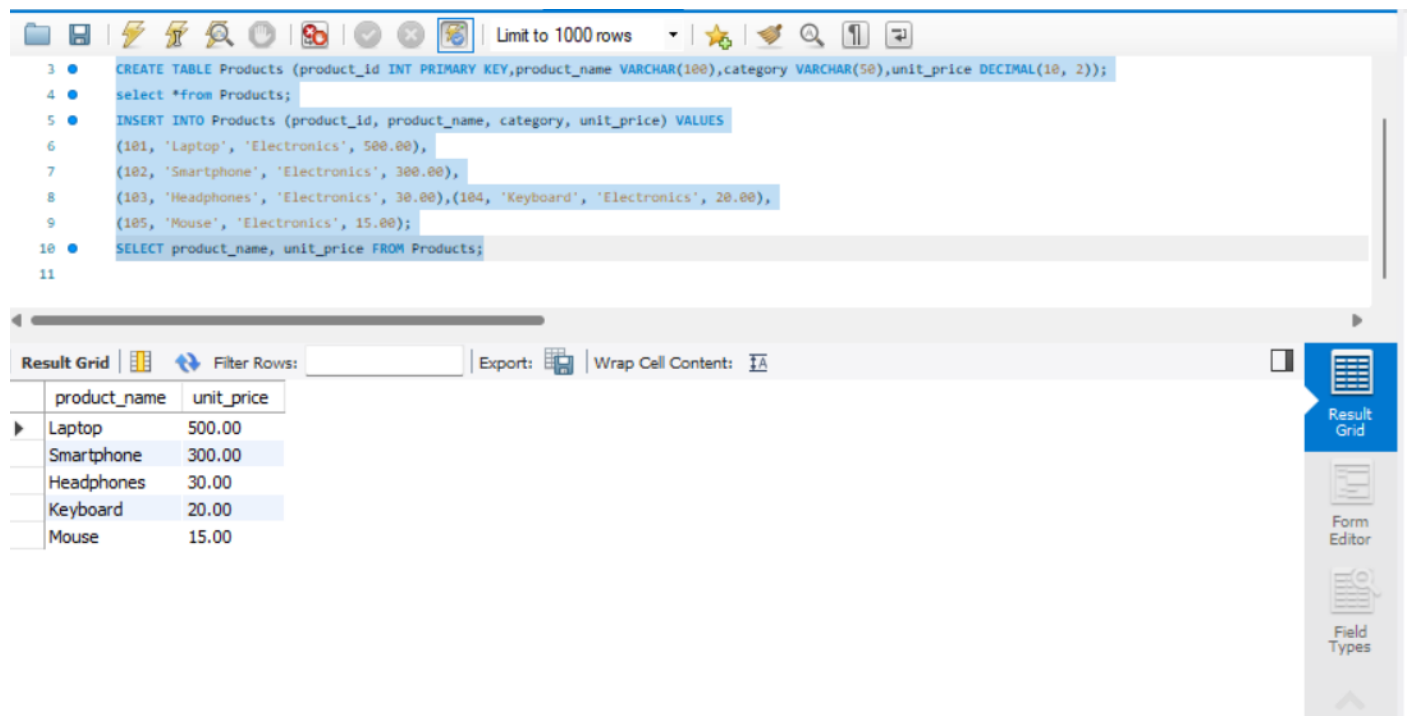
```
(102, 'Smartphone', 'Electronics', 300.00),
```

```
(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
SELECT product_name, unit_price FROM Products;
```

QUERY WITH OUTPUT:



The screenshot displays a database query editor interface. The top toolbar includes icons for file operations, execution, and search. The SQL editor contains the following code:

```
3 CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 select *from Products;
5 INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
6 (101, 'Laptop', 'Electronics', 500.00),
7 (102, 'Smartphone', 'Electronics', 300.00),
8 (103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
9 (105, 'Mouse', 'Electronics', 15.00);
10 SELECT product_name, unit_price FROM Products;
11
```

Below the editor, the 'Result Grid' tab is active, showing the output of the query. The grid has two columns: 'product_name' and 'unit_price'. The data is as follows:

product_name	unit_price
Laptop	500.00
Smartphone	300.00
Headphones	30.00
Keyboard	20.00
Mouse	15.00

On the right side of the interface, there are additional toolbars for 'Result Grid', 'Form Editor', and 'Field Types'.

3. Filter the Products table to show only products in the Electronics category.

QUERY:

create database ganesh;

use ganesh;

CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));

select *from Products;

INSERT INTO Products (product_id, product_name, category, unit_price) VALUES

(101, 'Laptop', 'Electronics', 500.00),

(102, 'Smartphone', 'Electronics', 300.00),

(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),

(105, 'Mouse', 'Electronics', 15.00);

SELECT * FROM Products WHERE category = 'Electronics';

QUERY WITH OUTPUT:

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 create database ganesh;
2 use ganesh;
3 CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 select *from Products;
5 INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
6 (101, 'Laptop', 'Electronics', 500.00),
7 (102, 'Smartphone', 'Electronics', 300.00),
8 (103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
9 (105, 'Mouse', 'Electronics', 15.00);
10 SELECT * FROM Products WHERE category = 'Electronics';
```

The result grid displays the output of the query, showing a table with 5 rows and 4 columns: product_id, product_name, category, and unit_price. The data is as follows:

product_id	product_name	category	unit_price
101	Laptop	Electronics	500.00
102	Smartphone	Electronics	300.00
103	Headphones	Electronics	30.00
104	Keyboard	Electronics	20.00
105	Mouse	Electronics	15.00

4. Retrieve the product_id and product_name from the Products table for products with a unit_price greater than \$100.

QUERY:

create database ganesh;

use ganesh;

CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));

select *from Products;

INSERT INTO Products (product_id, product_name, category, unit_price) VALUES

(101, 'Laptop', 'Electronics', 500.00),

(102, 'Smartphone', 'Electronics', 300.00),

(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),

(105, 'Mouse', 'Electronics', 15.00);

SELECT product_id, product_name FROM Products WHERE unit_price > 100;

QUERY WITH OUTPUT:

The screenshot displays the MySQL Workbench interface. The 'Query' tab is active, showing a SQL script that creates a 'Products' table and inserts five records. The 'Result Grid' shows the output of the final query, which selects product names and prices for products where the unit price is greater than 100. The 'Output' tab at the bottom shows the execution log, indicating that 2 rows were returned for the final query.

Query 1:

```
3 CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 select *from Products;
5 INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
6 (101, 'Laptop', 'Electronics', 500.00),
7 (102, 'Smartphone', 'Electronics', 300.00),
8 (103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
9 (105, 'Mouse', 'Electronics', 15.00);
10 SELECT product_id, product_name FROM Products WHERE unit_price > 100;
```

Result Grid:

product_id	product_name
101	Laptop
102	Smartphone

Output:

#	Time	Action	Message	Duration / Fetch
7	14:53:40	INSERT INTO Products (product_id, product_name, category, unit_price) VALUES (101, 'Laptop', 'Electronics', 500.00), (102, 'Smartphone', 'Electronics', 300.00), (103, 'Headphones', 'Electronics', 30.00), (104, 'Keyboard', 'Electronics', 20.00), (105, 'Mouse', 'Electronics', 15.00);	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.031 sec
8	14:53:47	select *from Products LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
9	14:54:49	select *from products LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
10	15:16:17	SELECT product_name, unit_price FROM Products LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
11	15:21:11	SELECT * FROM Products WHERE category = 'Electronics' LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec
12	15:25:07	SELECT product_id, product_name FROM Products WHERE unit_price > 100 LIMIT 0, 1000	2 row(s) returned	0.016 sec / 0.000 sec

5. Calculate the average unit_price of products in the Products table.

QUERY:

create database ganesh;

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
```

```
(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),
```

```
(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
SELECT AVG(unit_price) AS average_price FROM Products;
```

QUERY WITH OUTPUT:

The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
3 • CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 • select *from Products;
5 • INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
6   (101, 'Laptop', 'Electronics', 500.00),
7   (102, 'Smartphone', 'Electronics', 300.00),
8   (103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
9   (105, 'Mouse', 'Electronics', 15.00);
10 • SELECT AVG(unit_price) AS average_price FROM Products;
```

The result grid shows the output of the query:

average_price
173.000000

The IDE interface includes a toolbar with various icons for file operations, a 'Limit to 1000 rows' dropdown, and a sidebar with options for 'Result Grid', 'Form Editor', and 'Field Types'.

6. Retrieve product name and unit price from the Products table with the Highest Unit Price

QUERY:

create database ganesh;

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category  
VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

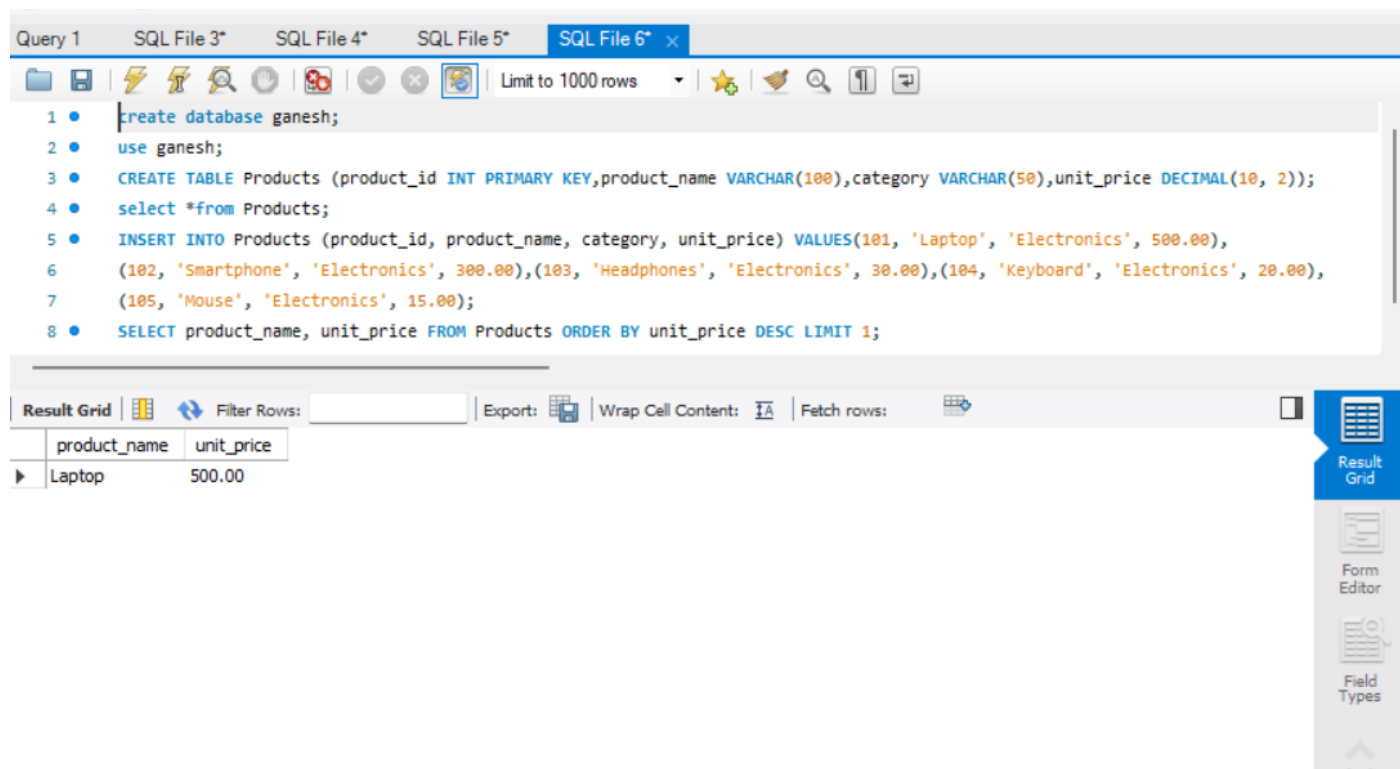
```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics',  
500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics',  
20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
SELECT product_name, unit_price FROM Products ORDER BY unit_price DESC LIMIT 1;
```

QUERY WITH OUTPUT:



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 • create database ganesh;  
2 • use ganesh;  
3 • CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));  
4 • select *from Products;  
5 • INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),  
6 • (102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),  
7 • (105, 'Mouse', 'Electronics', 15.00);  
8 • SELECT product_name, unit_price FROM Products ORDER BY unit_price DESC LIMIT 1;
```

The result grid shows the output of the query:

product_name	unit_price
Laptop	500.00

7. Retrieve the product_name and unit_price from the Products table, ordering the results by unit_price in descending order.

QUERY:

create database ganesh;

use ganesh;

CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));

select *from Products;

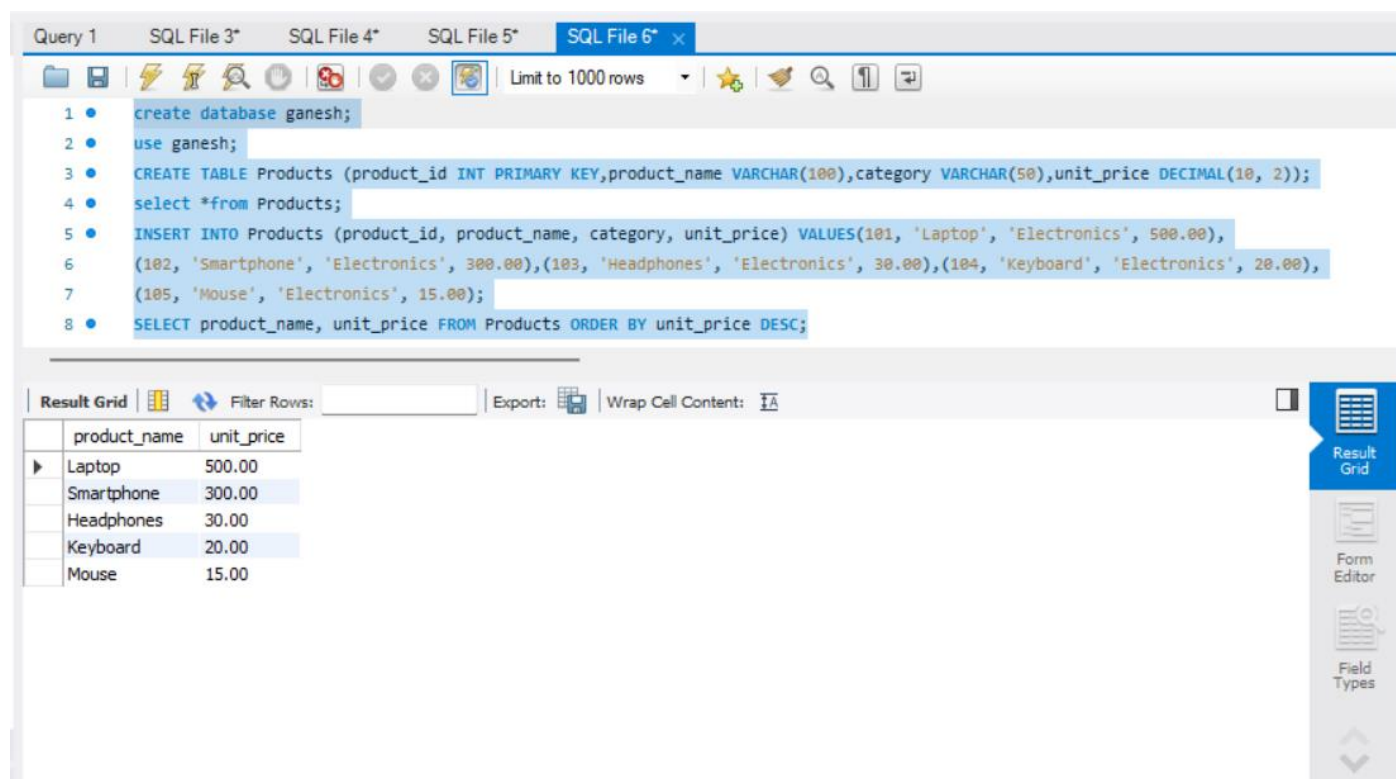
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),

(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),

(105, 'Mouse', 'Electronics', 15.00);

SELECT product_name, unit_price FROM Products ORDER BY unit_price DESC;

QUERY WITH OUTPUT:



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 • create database ganesh;
2 • use ganesh;
3 • CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 • select *from Products;
5 • INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
6 • (102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
7 • (105, 'Mouse', 'Electronics', 15.00);
8 • SELECT product_name, unit_price FROM Products ORDER BY unit_price DESC;
```

The result grid displays the output of the query, showing the product names and their unit prices in descending order:

product_name	unit_price
Laptop	500.00
Smartphone	300.00
Headphones	30.00
Keyboard	20.00
Mouse	15.00

8. Retrieve the product_name and unit_price from the Products table, filtering the unit_price to show only values between \$20 and \$600.

QUERY:

create database ganesh;

use ganesh;

CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));

select *from Products;

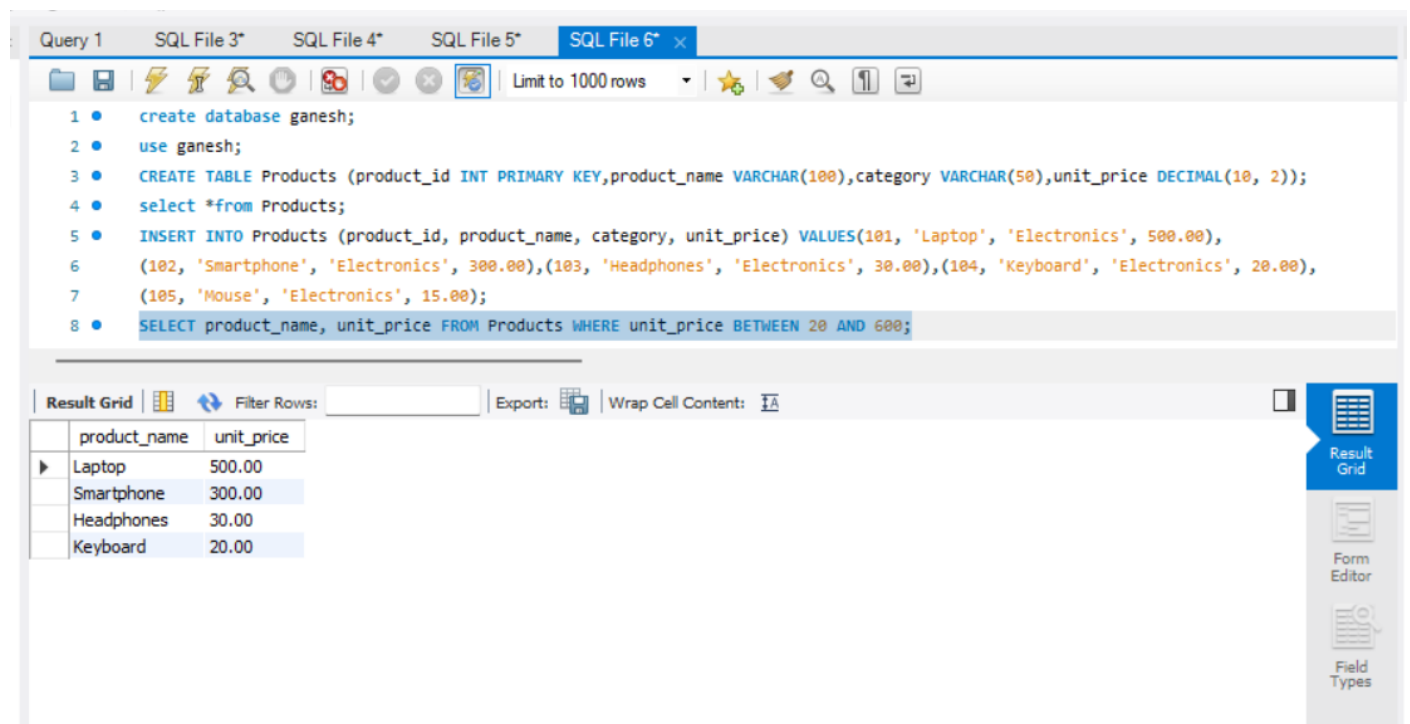
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),

(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),

(105, 'Mouse', 'Electronics', 15.00);

SELECT product_name, unit_price FROM Products WHERE unit_price BETWEEN 20 AND 600;

QUERY WITH OUTPUT:



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 • create database ganesh;
2 • use ganesh;
3 • CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 • select *from Products;
5 • INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
6 • (102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
7 • (105, 'Mouse', 'Electronics', 15.00);
8 • SELECT product_name, unit_price FROM Products WHERE unit_price BETWEEN 20 AND 600;
```

The result grid displays the output of the query, showing the product names and their unit prices. The data is as follows:

product_name	unit_price
Laptop	500.00
Smartphone	300.00
Headphones	30.00
Keyboard	20.00

9. Retrieve the product name and category from the Products table, ordering the results by category in ascending order.

QUERY:

create database ganesh;

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

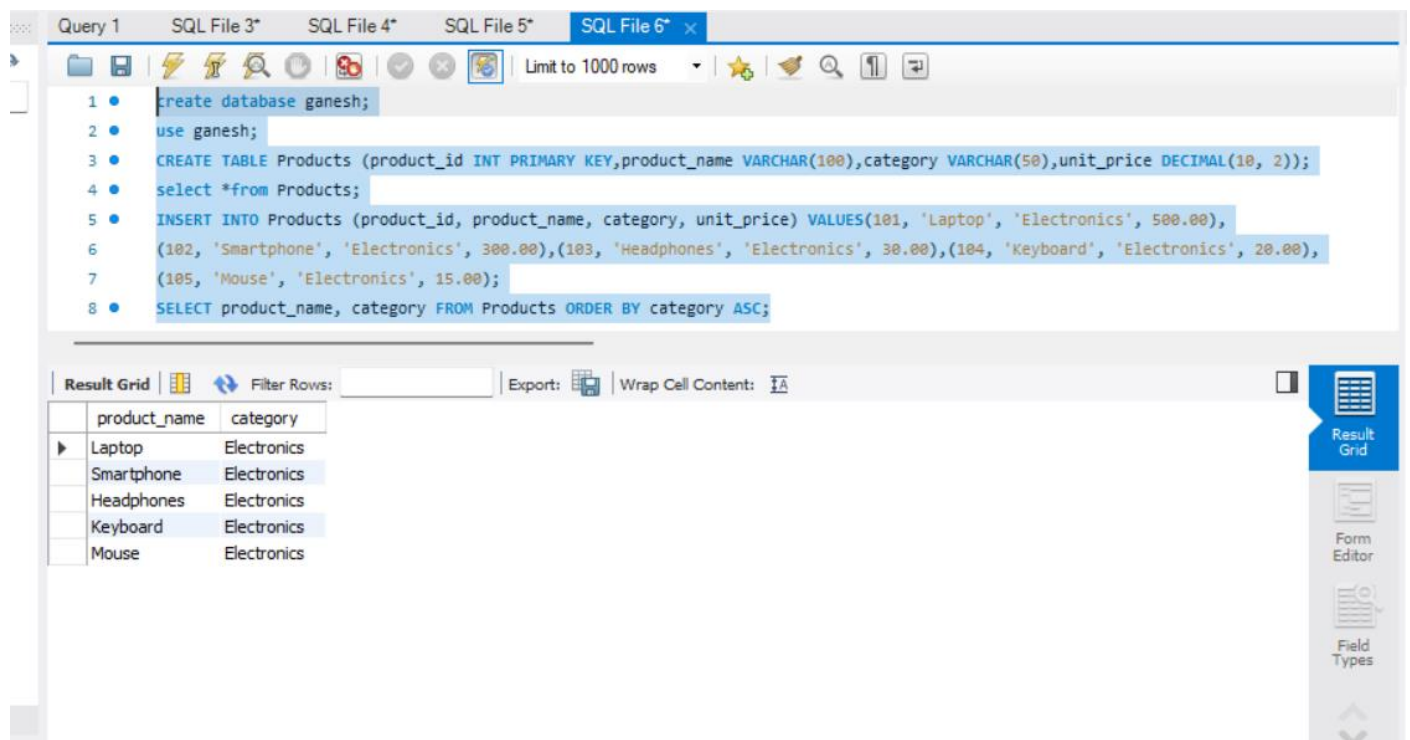
```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
SELECT product_name, category FROM Products ORDER BY category ASC;
```

QUERY WITH OUTPUT:



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query editor contains the following SQL code:

```
1 • create database ganesh;
2 • use ganesh;
3 • CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
4 • select *from Products;
5 • INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
6 • (102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
7 • (105, 'Mouse', 'Electronics', 15.00);
8 • SELECT product_name, category FROM Products ORDER BY category ASC;
```

The result grid displays the output of the query, showing a table with two columns: product_name and category. The data is sorted by category in ascending order.

product_name	category
Laptop	Electronics
Smartphone	Electronics
Headphones	Electronics
Keyboard	Electronics
Mouse	Electronics

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.

1. Retrieve all columns from the Sales table.

QUERY:

create database ganesh;

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

```
(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);
```

QUERY WITH OUTPUT:

Result Grid

Filter Rows:

Edit:

	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

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

QUERY:

create database ganesh;

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

```
(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);
```

```
SELECT sale_id, sale_date FROM Sales;
```

QUERY WITH OUTPUT:



The screenshot shows a 'Result Grid' with a toolbar at the top containing a grid icon, a refresh icon, and a 'Filter Rows:' label. The grid displays the output of the query 'SELECT sale_id, sale_date FROM Sales;'. It has two columns: 'sale_id' and 'sale_date'. The data rows are: (1, 2024-01-01), (2, 2024-01-02), (3, 2024-01-02), (4, 2024-01-03), and (5, 2024-01-03). A final row with 'NULL' values is also present. The grid has a scrollable area on the left with a play button icon.

	sale_id	sale_date
▶	1	2024-01-01
	2	2024-01-02
	3	2024-01-02
	4	2024-01-03
	5	2024-01-03
⚙	NULL	NULL

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

QUERY:

```
create database ganesh;
```

```
use ganesh;
```

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

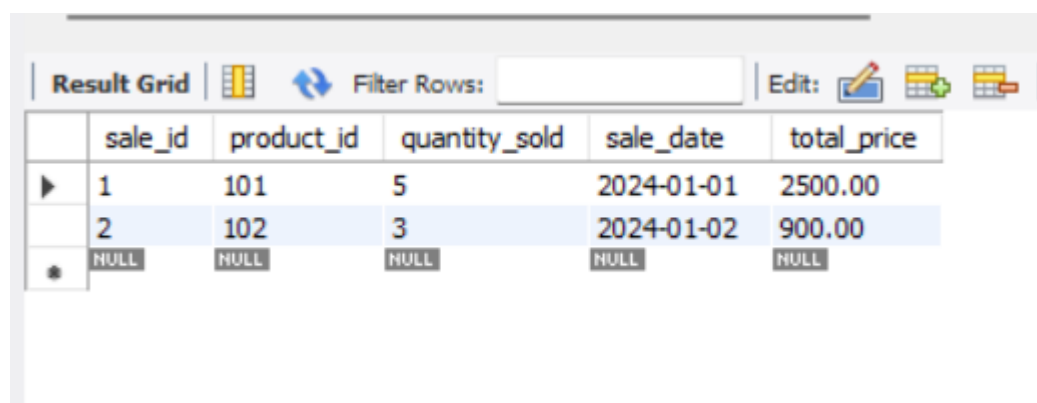
```
(1, 101, 5, '2024-01-01', 2500.00),(2, 102, 3, '2024-01-02', 900.00),(3, 103, 2, '2024-01-02', 60.00),
```

```
(4, 104, 4, '2024-01-03', 80.00),(5, 105, 6, '2024-01-03', 90.00);
```

```
SELECT * FROM Sales
```

```
WHERE total_price > 100;
```

QUERY WITH OUTPUT:



The screenshot shows a database query result grid with the following columns: sale_id, product_id, quantity_sold, sale_date, and total_price. The grid displays three rows of data, with the first two rows highlighted in blue. The third row is marked with a star icon and has NULL values for all columns.

	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
★	NULL	NULL	NULL	NULL	NULL

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

QUERY:

```
create database ganesh;
```

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

```
(1, 101, 5, '2024-01-01', 2500.00),(2, 102, 3, '2024-01-02', 900.00),(3, 103, 2, '2024-01-02', 60.00),
```

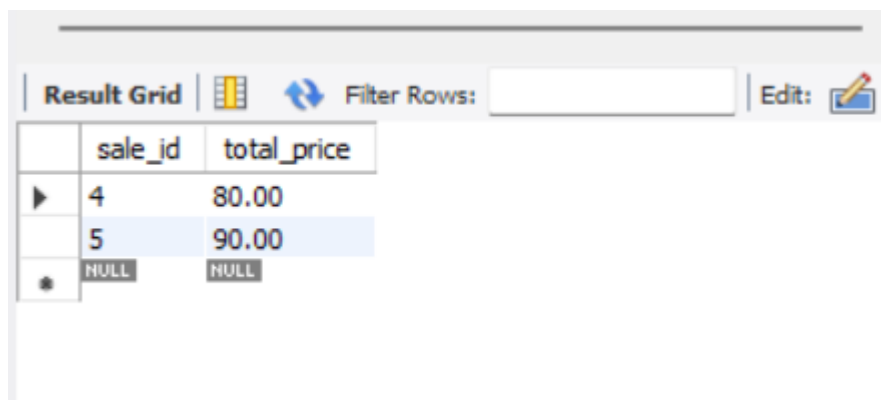
```
(4, 104, 4, '2024-01-03', 80.00),(5, 105, 6, '2024-01-03', 90.00);
```

```
SELECT sale_id, total_price
```

```
FROM Sales
```

```
WHERE sale_date = '2024-01-03';
```

QUERY WITH OUTPUT:



	sale_id	total_price
▶	4	80.00
	5	90.00
*	NULL	NULL

5. Calculate the total revenue generated from all sales in the Sales table.

QUERY:

```
create database ganesh;
```

use ganesh;

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category  
VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics',  
500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics',  
20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign  
key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

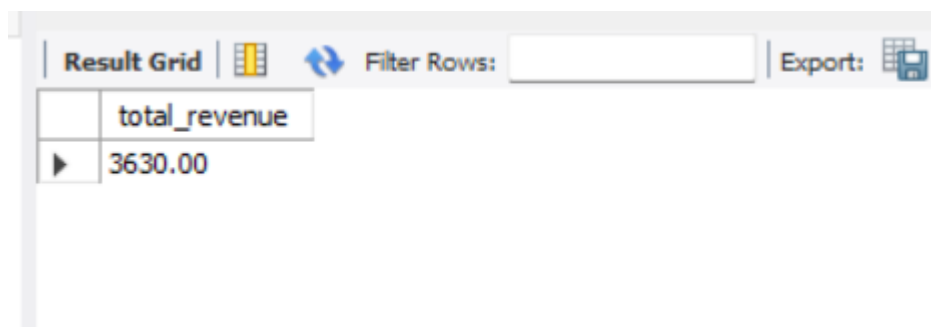
```
(1, 101, 5, '2024-01-01', 2500.00),(2, 102, 3, '2024-01-02', 900.00),(3, 103, 2, '2024-01-02', 60.00),
```

```
(4, 104, 4, '2024-01-03', 80.00),(5, 105, 6, '2024-01-03', 90.00);
```

```
SELECT SUM(total_price) AS total_revenue
```

```
FROM Sales;
```

QUERY WITH OUTPUT:



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains one row with the column 'total_revenue' and the value '3630.00'. Above the grid, there is a 'Filter Rows' input field and an 'Export' button. The grid has a small arrow icon next to the value '3630.00'.

total_revenue
3630.00

6. Calculate the total quantity_sold from the Sales table.

QUERY:

```
create database ganesh;
```

```
use ganesh;
```

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category  
VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics',  
500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics',  
20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign  
key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

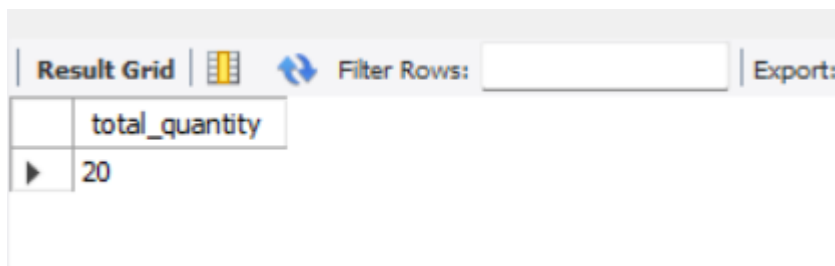
```
(1, 101, 5, '2024-01-01', 2500.00),(2, 102, 3, '2024-01-02', 900.00),(3, 103, 2, '2024-01-02', 60.00),
```

```
(4, 104, 4, '2024-01-03', 80.00),(5, 105, 6, '2024-01-03', 90.00);
```

```
SELECT SUM(quantity_sold) AS total_quantity
```

```
FROM Sales;
```

QUERY WITH OUTPUT:



The screenshot shows a database interface with a 'Result Grid' tab. The grid has two columns: 'total_quantity' and a value '20'. The interface also includes a 'Filter Rows' search bar and an 'Export' button.

	total_quantity
▶	20

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

QUERY:

```
create database ganesh;
```

```
use ganesh;
```

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```

```
INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics', 500.00),
```

```
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics', 20.00),
```

```
(105, 'Mouse', 'Electronics', 15.00);
```

```
CREATE TABLE Sales (
```

```
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign key (product_id) REFERENCES
```

```
Products(product_id));
```

```
select *from sales;
```

```
INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
```

```
(1, 101, 5, '2024-01-01', 2500.00),(2, 102, 3, '2024-01-02', 900.00),(3, 103, 2, '2024-01-02', 60.00),
```

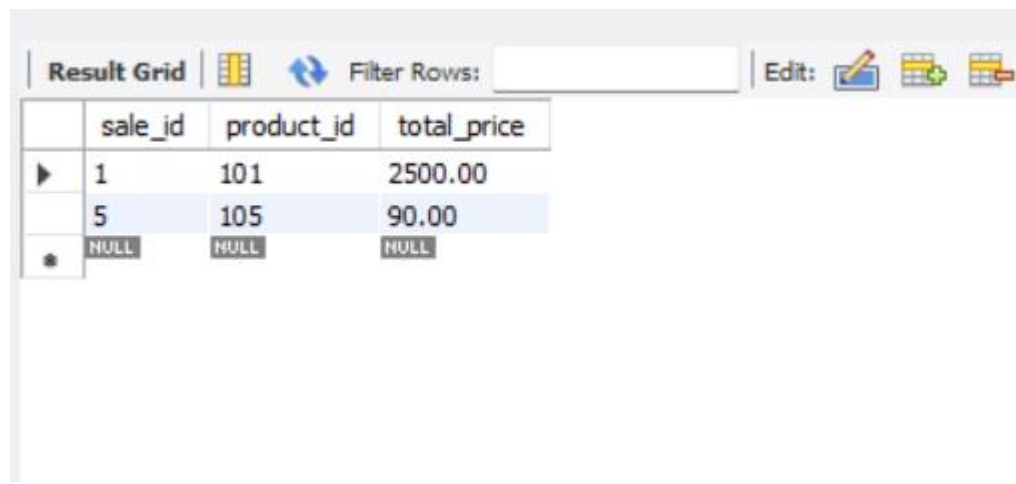
```
(4, 104, 4, '2024-01-03', 80.00),(5, 105, 6, '2024-01-03', 90.00);
```

```
SELECT sale_id, product_id, total_price
```

```
FROM Sales
```

```
WHERE quantity_sold > 4;
```

QUERY WITH OUTPUT:



	sale_id	product_id	total_price
▶	1	101	2500.00
	5	105	90.00
✱	NULL	NULL	NULL

8. Calculate the average total price of sales in the Sales table.

QUERY:

```
create database ganesh;
```

```
use ganesh;
```

```
CREATE TABLE Products (product_id INT PRIMARY KEY,product_name VARCHAR(100),category VARCHAR(50),unit_price DECIMAL(10, 2));
```

```
select *from Products;
```



```

INSERT INTO Products (product_id, product_name, category, unit_price) VALUES(101, 'Laptop', 'Electronics',
500.00),
(102, 'Smartphone', 'Electronics', 300.00),(103, 'Headphones', 'Electronics', 30.00),(104, 'Keyboard', 'Electronics',
20.00),
(105, 'Mouse', 'Electronics', 15.00);

CREATE TABLE Sales (
sale_id INT PRIMARY KEY, product_id INT, quantity_sold INT, sale_date DATE, total_price DECIMAL(10, 2) ,foreign
key (product_id) REFERENCES
Products(product_id));

select *from sales;

INSERT INTO Sales (sale_id, product_id, quantity_sold, sale_date, total_price) VALUES
(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);

SELECT AVG(total_price) AS average_price
FROM Sales;

```

QUERY WITH OUTPUT:

Result Grid			Filter Rows: <input type="text"/>	Export:
	average_price			
▶	726.000000			