Screenshots of the SQL Queries

```sql
select * from sales_data
where orderid >1005;
```

Data Output Messages Notifications

	orderid integer	customername text	product text	category text	quantity integer	price numeric (10,2)	orderdate date	region text
1	1006	Linda Martinez	Printer	Electronics	1	150.00	2024-01-12	South
2	1007	Michael Garcia	Bookshelf	Furniture	2	90.00	2024-01-13	East
3	1008	Barbara Anderson	Tablet	Electronics	1	300.00	2024-01-14	West
4	1009	William Thomas	Monitor	Electronics	2	180.00	2024-01-15	North
5	1010	Elizabeth Taylor	Mouse	Electronics	5	20.00	2024-01-16	South

The SQL query selects all columns from the sales_data table using SELECT * FROM. The WHERE clause filters rows where orderid is greater than 1005, showing only matching records. The semicolon marks the end of the statement. This helps retrieve specific data based on given conditions.

```sql

SELECT region, SUM(price) AS total_sales
FROM sales_data
GROUP BY region;
```

Data Output Messages Notifications

	region text	total_sales numeric
1	East	210.00
2	South	670.00
3	West	500.00
4	North	1010.00

The SQL query retrieves each region and calculates its total sales using SUM(price), renaming it as total_sales. Data is grouped by region with GROUP BY, so totals are shown per region. The result lists East, South, West, and North with their respective total sales values.

```sql
select * from sales_data
order by orderid asc;
```

Data Output Messages Notifications

	orderid integer	customermame text	product text	category text	quantity integer	price numeric (10,2)	orderdate date	region text
1	1001	Zeeshan Ansari	Laptop	Electronics	1	500.00	2024-01-05	North
2	1002	Mary Johnson	Smartphone	Electronics	2	600.00	2024-01-06	South
3	1003	James Brown	Office Chair	Furniture	1	150.00	2024-01-07	East
4	1004	Patricia Davis	Desk	Furniture	1	300.00	2024-01-08	West
5	1005	Robert Wilson	Headphones	Electronics	3	120.00	2024-01-10	North
6	1006	Linda Martinez	Printer	Electronics	1	200.00	2024-01-12	South
7	1007	Michael Garcia	Bookshelf	Furniture	2	100.00	2024-01-13	East

The SQL query selects all columns from sales_data and orders results by orderid in ascending order. The output shows details like order ID, customer name, product, category, quantity, price, order date, and region, listing sales records from various customers across different products and regions.

```sql
select * from sales_data as s
inner join customer as c
on s.customermame= c.customermame;
```

Data Output Messages Notifications

	orderid integer	customermame text	product text	category text	quantity integer	price numeric (10,2)	orderdate date	region text	customermame character varying (250)
1	1005	Robert Wilson	Headphones	Electronics	3	120.00	2024-01-10	North	Robert Wilson
2	1007	Michael Garcia	Bookshelf	Furniture	2	100.00	2024-01-13	East	Michael Garcia
3	1001	Zeeshan Ansari	Laptop	Electronics	1	500.00	2024-01-05	North	Zeeshan Ansari

The SQL query joins sales_data and customer tables using customermame and retrieves matching records. The result shows order details including ID, customer name, product, category, quantity, price, date, and region for three customers: Robert Wilson, Michael Garcia, and Zeeshan Ansari.

```sql
select * from sales_data as s
left join customer as c
on s.customername=c.customername;
```

Data Output Messages Notifications

	orderid integer	customername text	product text	category text	quantity integer	price numeric (10,2)	orderdate date	region text	customername character varying (250)
4	1005	Robert Wilson	Headphones	Electronics	3	120.00	2024-01-10	North	Robert Wilson
5	1006	Linda Martinez	Printer	Electronics	1	200.00	2024-01-12	South	[null]
6	1007	Michael Garcia	Bookshelf	Furniture	2	100.00	2024-01-13	East	Michael Garcia
7	1008	Barbara Anderson	Tablet	Electronics	1	340.00	2024-01-14	West	[null]
8	1009	William Thomas	Monitor	Electronics	2	200.00	2024-01-15	North	[null]
9	1010	Elizabeth Taylor	Mouse	Electronics	5	50.00	2024-01-16	South	[null]
10	1001	Zeeshan Ansari	Laptop	Electronics	1	500.00	2024-01-05	North	Zeeshan Ansari

The SQL query uses a LEFT JOIN between sales_data (alias s) and customer (alias c) on customername. The output lists order details with some matching customer names and some NULL values from the customer table when no match exists.

```sql
select * from sales_data as s
right join customer as c
on s.customername=c.customername;
```

Data Output Messages Notifications

	orderid integer	customername text	product text	category text	quantity integer	price numeric (10,2)	orderdate date	region text	customername character varying (250)
1	1005	Robert Wilson	Headphones	Electronics	3	120.00	2024-01-10	North	Robert Wilson
2	1007	Michael Garcia	Bookshelf	Furniture	2	100.00	2024-01-13	East	Michael Garcia
3	1001	Zeeshan Ansari	Laptop	Electronics	1	500.00	2024-01-05	North	Zeeshan Ansari
4	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	John Smith

The query performs a RIGHT JOIN between sales_data and customer on customername, returning all customers and matching sales records. If a customer has no sales data, sales columns show NULL. Here, John Smith appears with NULL values, indicating no matching sales record.

```sql
SELECT customername, price
FROM sales_data
WHERE price > (
    SELECT AVG(price) FROM sales_data
);
```

Data Output Messages Notifications

	customername text	price numeric (10,2)
1	Mary Johnson	600.00
2	Patricia Davis	300.00
3	Barbara Anderson	340.00
4	Zeeshan Ansari	500.00

The SQL query selects customername and price from sales_data where the price is greater than the average price of all sales. The result shows four customers—Mary Johnson, Patricia Davis, Barbara Anderson, and Zeeshan Ansari—with prices above the dataset's average.

```sql
select round(avg(price),2) from sales_data;
```

Data Output Messages Notifications

	round numeric
1	256.00

The SQL query calculates the average price from the sales_data table, rounds it to two decimal places, and returns the result as 256.00. This value represents the mean price of all entries in the dataset.

```
select sum(price) from sales_data;
```

Data Output Messages Notifications

	sum numeric
1	2560.00

The SQL query calculates the total sum of the price column from the sales_data table. The result, 2560.00, represents the combined total of all price values in the dataset.

```
CREATE VIEW region_sales AS
SELECT region, SUM(price) AS total_sales
FROM sales_data
GROUP BY region;

SELECT * FROM region_sales;
```

Data Output Messages Notifications

	region text	total_sales numeric
1	East	250.00
2	South	850.00
3	West	640.00
4	North	820.00

The SQL code creates a view region_sales that groups data by region from sales_data and calculates total sales using SUM(price). The output shows: East - 250.00, South - 850.00, West - 640.00, and North - 820.00, representing total sales for each region.