a.Use SELECT, WHERE, ORDER BY, GROUP BY

```sql
-- Select customers who signed up after Jan 1, 2023:
SELECT * FROM customers WHERE created_at > '2023-01-01' ORDER BY created_at DESC;
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| customer_id | first_name | last_name | email | phone | created_at |
|---|---|---|---|---|---|
| 2 | Bob | Smith | bob@example.com | 2345678901 | 2023-02-15 |
| 1 | Alice | Johnson | alice@example.com | 1234567890 | 2023-01-10 |
| NULL | NULL | NULL | NULL | NULL | NULL |

```sql
-- Total sales per product:
SELECT
    p.product_name,
    SUM(oi.quantity) AS total_quantity_sold,
    SUM(oi.price * oi.quantity) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sales DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| product_name | total_quantity_sold | total_sales |
|---|---|---|
| Laptop | 1 | 1200.00 |
| Smartphone | 1 | 500.00 |
| Keyboard | 2 | 200.00 |

b.Use JOINS (INNER, LEFT, RIGHT)

```sql
-- INNER JOIN: Orders with customer info
SELECT
    o.order_id,
    c.first_name,
    c.last_name,
    o.order_date,
    o.status
FROM orders o
INNER JOIN customers c ON o.customer_id = c.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| order_id | first_name | last_name | order_date | status |
|---|---|---|---|---|
| 1001 | Alice | Johnson | 2023-03-01 | Shipped |
| 1002 | Bob | Smith | 2023-03-05 | Delivered |

```sql
-- LEFT JOIN: All customers with their orders (even those without orders)
SELECT
    c.customer_id,
    c.first_name,
    o.order_id,
    o.status
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| customer_id | first_name | order_id | status |
|---|---|---|---|
| 1 | Alice | 1001 | Shipped |
| 2 | Bob | 1002 | Delivered |

```sql
-- RIGHT JOIN: All orders including those without matching customer
SELECT
    o.order_id,
    o.customer_id,
    c.first_name
FROM orders o
RIGHT JOIN customers c ON o.customer_id = c.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| order_id | customer_id | first_name |
|---|---|---|
| 1001 | 1 | Alice |
| 1002 | 2 | Bob |

c.Write subqueries

```sql
-- Find customers who made payments over $1000 total
SELECT customer_id, first_name, last_name
FROM customers
WHERE customer_id IN (
    SELECT o.customer_id
    FROM orders o
    JOIN payments p ON o.order_id = p.order_id
    GROUP BY o.customer_id
    HAVING SUM(p.amount) > 1000
);
```

Result Grid | Filter Rows: | Edit: | Export/Impor

| customer_id | first_name | last_name |
|---|---|---|
| 2 | Bob | Smith |
| NULL | NULL | NULL |

d.Use aggregate functions (SUM, AVG)

```
-- Average and Sum  product price
SELECT AVG(price) AS avg_price , sum(price) as Sum_Price  FROM products;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| avg_price | Sum_Price |
|---|---|
| 600.000000 | 1800.00 |

e. CREATE VIEWS FOR ANALYSIS

```
-- Create view Average and Sum  product price
Create View  Average_and_Sum_product_price as
SELECT AVG(price) AS avg_price , sum(price) as Sum_Price  FROM products;
```

f.Optimize queries with indexes

```
-- Index for customer_id in orders
CREATE INDEX idx_orders_customer_id ON orders(customer_id);
```