Exploratory Data Analysis

-- 1. Basic Queries

```
-- SELECT: The `SELECT` statement is used to retrieve data from one or more tables.
-- SELECT all columns
SELECT * FROM customers LIMIT 5;

-- SELECT specific columns with ALIAS
SELECT
    product_name AS 'Product Name',
    purchase_price AS 'Cost Price'
FROM
    products
LIMIT 10;
```

Product Name	Cost Price
Silk Blend Blouse	85.5
Urban Runner Sneakers	120
Organic Cotton Jeans	65
Nordic Wool Peacoat	250
Tokyo Graphic Tee	45
Parisian Midi Dress	180
London Plaid Belt	35
Berlin Cashmere Wallet	25
Aussie Outback Boots	95
Seoul Leather Handbag	220

#Average Product price:

```
-- AVG
SELECT
    AVG(purchase_price) AS avg_product_cost
FROM
    products;
```

	avg_product_cost
▶	112.050000

-- Find customers who placed orders after 2023-06-01 AND have spent more than $500 in total:

```
SELECT
   customer_id,
   SUM(total_amount) AS total_spent,
   COUNT(order_id) AS num_orders
FROM
   sales_orders
WHERE
   order_date > '2023-06-01'
GROUP BY customer_id
HAVING SUM(total_amount) > 500.00; -- Filters groups AFTER aggregation
```

	customer_id	total_spent	num_orders
▶	1	509.97	3

-- 4. Built-in Functions for Data Manipulation & Analysis

```
-- Date/Time Functions (MySQL Equivalents)
-- EXTRACT / DATE_PART -> YEAR(), MONTH(), DAY(), HOUR(), EXTRACT()

SELECT
   order_date,
   YEAR(order_date) AS order_year,
   MONTH(order_date) AS order_month
FROM
   sales_orders
LIMIT 5;



SELECT
   order_date,
   EXTRACT(QUARTER FROM order_date) AS order_quarter
FROM
   sales_orders
LIMIT 5;
```

	order_date	order_quarter
▶	2023-06-15 10:30:00	2
	2023-06-20 14:00:00	2
	2023-07-05 09:00:00	3
	2023-07-10 11:45:00	3
	2023-08-01 16:20:00	3

-- Example: Calculate discount percentage, avoid division by zero if selling_price is 0:

```
SELECT
    order_item_id,
    selling_price,
    discount_amount,
    (discount_amount * 100 / NULLIF(selling_price, 0)) AS
discount_percentage
FROM
    order_items
LIMIT 10;
```

– -- TO_CHAR -> DATE_FORMAT()

```
SELECT
    order_date,
    DATE_FORMAT(order_date, '%Y/%m/%d %H:%i') AS formatted_date
FROM
    sales_orders
LIMIT 5;
```

-- TO_DATE -> STR_TO_DATE()

```
SELECT STR_TO_DATE("2024-May-24", "%Y-%b-%d") as Date;
```

```
─SELECT
    order_id,
    CONCAT('Order #', CAST(order_id AS CHAR)) AS order_label
FROM
    sales_orders
LIMIT 5;
```

-- POSITION / LOCATE

```
SELECT
    email, LOCATE('@', email) AS at_position
FROM
    customers
LIMIT 5;
```

email	at_position
alice.s@example.com	8
bob.j@example.com	6
charlie.w@example.com	10
diana.b@example.com	8
ethan.j@example.com	8

```
SELECT
    order_id,
    CONCAT('Order #', CAST(order_id AS CHAR)) AS order_label
FROM
    sales_orders
LIMIT 5;
```

-- SUBSTRING

```
SELECT
    product_name, SUBSTRING(product_name, 1, 10) AS name_start
FROM
    products
LIMIT 5;
```

product_name	name_start
Silk Blend Blouse	Silk Blend
Urban Runner Sneakers	Urban Runn
Organic Cotton Jeans	Organic Co
Nordic Wool Peacoat	Nordic Woo
Tokyo Graphic Tee	Tokyo Grap

-- Number of customers

```
SELECT COUNT(*) AS total_customers FROM customers;
```

	total_customers
▶	10

-- Number of orders by status

```
SELECT status, COUNT(*) FROM sales_orders GROUP BY status;
```

-- Most popular product categories

```
SELECT c.category_name, COUNT(*) AS total_orders
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY total_orders DESC;
```