```sql
select * from INFORMATION_SCHEMA.tables
/*
===============================================================================
Database Exploration
===============================================================================
Purpose:
    - To explore the structure of the database, including the list of tables and
      their schemas.
    - To inspect the columns and metadata for specific tables.

Table Used:
    - INFORMATION_SCHEMA.TABLES
    - INFORMATION_SCHEMA.COLUMNS
===============================================================================
*/

-- Retrieve a list of all tables in the database
SELECT
    TABLE_CATALOG,
    TABLE_SCHEMA,
    TABLE_NAME,
    TABLE_TYPE
FROM INFORMATION_SCHEMA.TABLES;

-- Retrieve all columns for a specific table (dim_customers)
SELECT
    COLUMN_NAME,
    DATA_TYPE,
    IS_NULLABLE,
    CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'dim_customers';


/*
===============================================================================
Dimensions Exploration
===============================================================================
Purpose:
    - To explore the structure of dimension tables.

SQL Functions Used:
    - DISTINCT
    - ORDER BY
===============================================================================
*/

-- Retrieve a list of unique countries from which customers originate
```

```sql
SELECT DISTINCT
    country
FROM gold.dim_customers
ORDER BY country;

-- Retrieve a list of unique categories, subcategories, and products
SELECT DISTINCT
    category,
    subcategory,
    product_name
FROM gold.dim_products
ORDER BY category, subcategory, product_name;

/*
===============================================================================
Date Range Exploration
===============================================================================
Purpose:
    - To determine the temporal boundaries of key data points.
    - To understand the range of historical data.

SQL Functions Used:
    - MIN(), MAX(), DATEDIFF()
===============================================================================
*/

-- Determine the first and last order date and the total duration in months
SELECT
    MIN(order_date) AS first_order_date,
    MAX(order_date) AS last_order_date,
    DATEDIFF(MONTH, MIN(order_date), MAX(order_date)) AS order_range_months
FROM gold.fact_sales;

-- Find the youngest and oldest customer based on birthdate
SELECT
    MIN(birthdate) AS oldest_birthdate,
    DATEDIFF(YEAR, MIN(birthdate), GETDATE()) AS oldest_age,
    MAX(birthdate) AS youngest_birthdate,
    DATEDIFF(YEAR, MAX(birthdate), GETDATE()) AS youngest_age
FROM gold.dim_customers;


/*
===============================================================================
Measures Exploration (Key Metrics)
===============================================================================
Purpose:
    - To calculate aggregated metrics (e.g., totals, averages) for quick insights.
    - To identify overall trends or spot anomalies.
```

```
SQL Functions Used:
    - COUNT(), SUM(), AVG()
===============================================================================
*/

-- Find the Total Sales
SELECT SUM(sales_amount) AS total_sales FROM gold.fact_sales

-- Find how many items are sold
SELECT SUM(quantity) AS total_quantity FROM gold.fact_sales

-- Find the average selling price
SELECT AVG(price) AS avg_price FROM gold.fact_sales

-- Find the Total number of Orders
SELECT COUNT(order_number) AS total_orders FROM gold.fact_sales
SELECT COUNT(DISTINCT order_number) AS total_orders FROM gold.fact_sales

-- Find the total number of products
SELECT COUNT(product_name) AS total_products FROM gold.dim_products

-- Find the total number of customers
SELECT COUNT(customer_key) AS total_customers FROM gold.dim_customers;

-- Find the total number of customers that has placed an order
SELECT COUNT(DISTINCT customer_key) AS total_customers FROM gold.fact_sales;

-- Generate a Report that shows all key metrics of the business
SELECT 'Total Sales' AS measure_name, SUM(sales_amount) AS measure_value FROM
  gold.fact_sales
UNION ALL
SELECT 'Total Quantity', SUM(quantity) FROM gold.fact_sales
UNION ALL
SELECT 'Average Price', AVG(price) FROM gold.fact_sales
UNION ALL
SELECT 'Total Orders', COUNT(DISTINCT order_number) FROM gold.fact_sales
UNION ALL
SELECT 'Total Products', COUNT(DISTINCT product_name) FROM gold.dim_products
UNION ALL
SELECT 'Total Customers', COUNT(customer_key) FROM gold.dim_customers;



/*
===============================================================================
Magnitude Analysis
===============================================================================
Purpose:
```

```
    - To quantify data and group results by specific dimensions.
    - For understanding data distribution across categories.

SQL Functions Used:
    - Aggregate Functions: SUM(), COUNT(), AVG()
    - GROUP BY, ORDER BY
===============================================================================
*/

-- Find total customers by countries
SELECT
    country,
    COUNT(customer_key) AS total_customers
FROM gold.dim_customers
GROUP BY country
ORDER BY total_customers DESC;

-- Find total customers by gender
SELECT
    gender,
    COUNT(customer_key) AS total_customers
FROM gold.dim_customers
GROUP BY gender
ORDER BY total_customers DESC;

-- Find total products by category
SELECT
    category,
    COUNT(product_key) AS total_products
FROM gold.dim_products
GROUP BY category
ORDER BY total_products DESC;

-- What is the average costs in each category?
SELECT
    category,
    AVG(cost) AS avg_cost
FROM gold.dim_products
GROUP BY category
ORDER BY avg_cost DESC;

-- What is the total revenue generated for each category?
SELECT
    p.category,
    SUM(f.sales_amount) AS total_revenue
FROM gold.fact_sales f
LEFT JOIN gold.dim_products p
    ON p.product_key = f.product_key
GROUP BY p.category
```

```sql
ORDER BY total_revenue DESC;


-- What is the total revenue generated by each customer?
SELECT
    c.customer_key,
    c.first_name,
    c.last_name,
    SUM(f.sales_amount) AS total_revenue
FROM gold.fact_sales f
LEFT JOIN gold.dim_customers c
    ON c.customer_key = f.customer_key
GROUP BY
    c.customer_key,
    c.first_name,
    c.last_name
ORDER BY total_revenue DESC;

-- What is the distribution of sold items across countries?
SELECT
    c.country,
    SUM(f.quantity) AS total_sold_items
FROM gold.fact_sales f
LEFT JOIN gold.dim_customers c
    ON c.customer_key = f.customer_key
GROUP BY c.country
ORDER BY total_sold_items DESC;

/*
===============================================================================
Ranking Analysis
===============================================================================
Purpose:
    - To rank items (e.g., products, customers) based on performance or other
      metrics.
    - To identify top performers or laggards.

SQL Functions Used:
    - Window Ranking Functions: RANK(), DENSE_RANK(), ROW_NUMBER(), TOP
    - Clauses: GROUP BY, ORDER BY
===============================================================================
*/

-- Which 5 products Generating the Highest Revenue?
-- Simple Ranking
SELECT TOP 5
    p.product_name,
    SUM(f.sales_amount) AS total_revenue
FROM gold.fact_sales f
```

```sql
LEFT JOIN gold.dim_products p
    ON p.product_key = f.product_key
GROUP BY p.product_name
ORDER BY total_revenue DESC;

-- Complex but Flexibly Ranking Using Window Functions
SELECT *
FROM (
    SELECT
        p.product_name,
        SUM(f.sales_amount) AS total_revenue,
        RANK() OVER (ORDER BY SUM(f.sales_amount) DESC) AS rank_products
    FROM gold.fact_sales f
    LEFT JOIN gold.dim_products p
        ON p.product_key = f.product_key
    GROUP BY p.product_name
) AS ranked_products
WHERE rank_products <= 5;

-- What are the 5 worst-performing products in terms of sales?
SELECT TOP 5
    p.product_name,
    SUM(f.sales_amount) AS total_revenue
FROM gold.fact_sales f
LEFT JOIN gold.dim_products p
    ON p.product_key = f.product_key
GROUP BY p.product_name
ORDER BY total_revenue;

-- Find the top 10 customers who have generated the highest revenue
SELECT TOP 10
    c.customer_key,
    c.first_name,
    c.last_name,
    SUM(f.sales_amount) AS total_revenue
FROM gold.fact_sales f
LEFT JOIN gold.dim_customers c
    ON c.customer_key = f.customer_key
GROUP BY
    c.customer_key,
    c.first_name,
    c.last_name
ORDER BY total_revenue DESC;

-- The 3 customers with the fewest orders placed
SELECT TOP 3
    c.customer_key,
    c.first_name,
    c.last_name,
```

```
    COUNT(DISTINCT order_number) AS total_orders
FROM gold.fact_sales f
LEFT JOIN gold.dim_customers c
    ON c.customer_key = f.customer_key
GROUP BY
    c.customer_key,
    c.first_name,
    c.last_name
ORDER BY total_orders ;
```