```sql
-- create database
create database sales;

use sales;


-- Data Loading:
-- 1. Import all the dimension tables using the Table Data Import Wizard.(DONE)
-- 2. Load the fact_sales table using the LOAD DATA INFILE method (bulk loading).
show variables like 'local_infile';

set global local_infile = 1;

use sales;

load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Fact Sales Data.csv'
into table fact_sales
fields terminated by ','
optionally enclosed by '"'
lines terminated by '\r\n'
ignore 1 rows
(OrderDate, StockDate, OrderNumber, ProductKey, CustomerKey, TerritoryKey, OrderLineItem, OrderQuantity);



-- Data Cleaning:
-- 1. Convert all date columns into proper DATE format.
-- column datatype change for calender
set sql_safe_updates =0;

update calendar
set date = str_to_date(date,'%m/%d/%Y');

alter table calendar
modify column date Date;

-- column datatype change for customer_lookup
update customer_lookup
set BirthDate = str_to_date(BirthDate,'%m/%d/%Y');

alter table customer_lookup
modify column BirthDate Date;

-- column datatype change for return_data
alter table returns_data
modify column ReturnDate Date;

-- column datatype change for fact_sales
-- OrderDate
update fact_sales
set OrderDate = str_to_date(OrderDate,'%m/%d/%Y');

alter table fact_sales
modify column OrderDate Date;

-- StockDate
update fact_sales
set StockDate = str_to_date(StockDate,'%m/%d/%Y');

alter table fact_sales
modify column StockDate Date;


-- 2. Ensure numeric fields are stored in correct numeric data types (INT, DECIMAL).
-- column datatype change for Product
alter table product
```

```sql
modify column ProductCost Decimal(10,2);

alter table product
modify column ProductPrice Decimal(10,2);




-- QUESTIONS

-- 1. Find total sales quantity per product.
select f.ProductKey, p.ProductName, sum(f.OrderQuantity) as TotalSalesQuantity
from fact_sales as f join product as p
on f.ProductKey = p.ProductKey
group by f.ProductKey, p.ProductName
order by TotalSalesQuantity desc;



-- 2. Show total sales revenue per region.
select t.Region, sum(p.ProductPrice * f.OrderQuantity) as TotalRevenue
from fact_sales as f
join product as p on p.ProductKey= f.ProductKey
join territory as t on t.SalesTerritoryKey = f.TerritoryKey
group by t.Region
order by TotalRevenue desc;



-- 3. Get total revenue per product category.
select pc.ProductCategoryKey, pc.CategoryName, sum(p.ProductPrice * f.OrderQuantity) as TotalRevenue_PerProduct
from fact_sales as f
join product as p on p.ProductKey = f.ProductKey
join product_subcategory as ps on ps.ProductSubcategoryKey = p.ProductSubcategoryKey
join product_category as pc on pc.ProductCategoryKey = ps.ProductCategoryKey
group by pc.ProductCategoryKey, pc.CategoryName
order by TotalRevenue_PerProduct desc;



-- 4. Find Top 10 customers who have spent the most.
select f.CustomerKey, cl.Full_name, sum(p.ProductPrice * OrderQuantity) as Total_spent
from fact_sales as f
join product as p on p.ProductKey = f.ProductKey
join customer_lookup as cl on cl.CustomerKey = f.CustomerKey
group by f.CustomerKey, cl.Full_name
order by Total_spent desc
limit 10;



-- 5. Get total orders by region.
select t.region, sum(f.OrderQuantity) as Total_Orders
from fact_sales as f
join territory as  t on t.SalesTerritoryKey = f.TerritoryKey
group by t.region
order by Total_Orders desc;


-- 6. Stored procedure to calculate total revenue for a given category.
delimiter $$
create procedure calculate_total_revenue(in category_key int)
begin
        select pc.ProductCategoryKey, pc.CategoryName, sum(p.ProductPrice * f.OrderQuantity) as Total_revenue
    from fact_sales as f
    join product as p on p.ProductKey = f.ProductKey
    join product_subcategory as ps on ps.ProductSubcategoryKey = p.ProductSubcategoryKey
```

```sql
    join product_category as pc on pc.ProductCategoryKey = ps.ProductCategoryKey
    where pc.ProductCategoryKey = category_key
    group by pc.CategoryName, pc.ProductCategoryKey
    order by Total_revenue desc;

end$$
delimiter ;

call calculate_total_revenue(1);



-- 7. Stored procedure to get products sold in a given date range.
delimiter //
create procedure products_sold_givenDate(in start_date date, in end_date date)
begin
        select p.ProductName, f.OrderDate, sum(f.OrderQuantity) as TotalSold
    from fact_sales as f
    join product as p on p.ProductKey = f.ProductKey
    where f.OrderDate between start_date and end_date
    group by p.ProductName, f.OrderDate;
end//
delimiter ;

call products_sold_givenDate('2020-01-02','2020-01-10');


-- 8. Compare each month’s sales with previous month using LAG().
with MonthlySales as (
    select year(OrderDate) as year,
           month(OrderDate) as month,
           sum(OrderQuantity) as Sales
    from fact_sales
    group by year(OrderDate), month(OrderDate)
)
select Year, Month, Sales,
       lag(Sales) over(order by Year, Month) as Pre_MonthSales
from MonthlySales
order by year asc, month asc;


-- 9. Number orders per customer with ROW_NUMBER().
with CustomerOrders as (
    select CustomerKey, count(*) as Orders_num
    from fact_sales
    group by CustomerKey
)
select CustomerKey, Orders_num,
       row_number() over(order by Orders_num desc) as Row_Num
from CustomerOrders
order by Row_Num;


-- 10. Find repeat customers.
select cl.CustomerKey, cl.Full_name, count(f.OrderQuantity) as Orders_repeat
from fact_sales as f
join customer_lookup cl on cl.CustomerKey = f.CustomerKey
group by cl.CustomerKey, cl.Full_name
having count(f.OrderQuantity) > 1
order by Orders_repeat desc;



-- 11. Find percentage of returned products
with Ordered as (
    select ProductKey, sum(OrderQuantity) as TotalOrdered
    from fact_sales
    group by ProductKey
```

```sql
),
Returned as (
    select ProductKey, sum(ReturnQuantity) as TotalReturned
    from returns_data
    group by ProductKey
)
select
    p.ProductKey,
    p.ProductName,
    r.TotalReturned,
    o.TotalOrdered,
    round((r.TotalReturned / o.TotalOrdered) * 100, 2) as Return_Percentage
from product p
join Ordered o on p.ProductKey = o.ProductKey
join Returned r on p.ProductKey = r.ProductKey
order by Return_Percentage desc;



-- 12. Most popular product in each category
with RankedProducts as (
    select pc.CategoryName, p.ProductName,
        sum(f.OrderQuantity) as TotalSales,
        rank() over(partition by pc.CategoryName order by sum(f.OrderQuantity) desc) as ProductRank
    from fact_sales f
    join product p on p.ProductKey = f.ProductKey
    join product_subcategory ps on ps.ProductSubcategoryKey = p.ProductSubcategoryKey
    join product_category pc on pc.ProductCategoryKey = ps.ProductCategoryKey
    group by pc.CategoryName, p.ProductName
)
select CategoryName, ProductName, TotalSales
from RankedProducts
where ProductRank = 1
order by CategoryName;


-- 13. Top spending customer per region.
with RankedCustomers as (
    select t.region, cl.CustomerKey, cl.Full_name,
        sum(f.OrderQuantity * p.ProductPrice) as Total_Spending,
        rank() over(partition by t.region order by sum(f.OrderQuantity * p.ProductPrice) desc) as SpendingRank
    from fact_sales f
    join product p on p.ProductKey = f.ProductKey
    join customer_lookup cl on cl.CustomerKey = f.CustomerKey
    join territory t on t.SalesTerritoryKey = f.TerritoryKey
    group by t.region, cl.CustomerKey, cl.Full_name
)
select region, CustomerKey, Full_name, Total_Spending
from RankedCustomers
where SpendingRank = 1
order by region;



-- 14. Difference between product’s price and average price in its category
select pc.CategoryName,p.ProductName,p.ProductPrice,
    p.ProductPrice - avg(p.ProductPrice) over (partition by pc.ProductCategoryKey) as Price_Difference
from product p
join product_subcategory ps on ps.ProductSubcategoryKey = p.ProductSubcategoryKey
join product_category pc on pc.ProductCategoryKey = ps.ProductCategoryKey;



-- 15. Customers who placed orders in multiple territories
select cl.CustomerKey,cl.Full_name,count(distinct f.TerritoryKey) as Terr_count
from fact_sales f
join customer_lookup cl on cl.CustomerKey = f.CustomerKey
```

```
271 group by cl.CustomerKey, cl.Full_name
272 having count(distinct f.TerritoryKey) > 1
273 order by Terr_count desc;
```