```
use revision
---------Case Study 1---------

/* PROBLEM STATEMENT

As a database administrator, you aim to leverage customer data to answer several
  key questions, particularly concerning sales and profit from different states,
  marketing expenditures,
and other metrics such as Cost of Goods Sold (COGS) and budget profit.
Your goal is to use these insights to determine the best-selling items.
Due to privacy concerns, you have only a sample of the overall customer data.
However, you believe this sample is sufficient to write effective SQL queries to
  answer your questions.

DATASETS:

FactTable
---------------
Columns: Date, ProductID, Profit, Sales, Margin, COGS, Total Expenses, Marketing,
  Inventory, Budget Profit, Budget COGS, Budget Margin, Budget Sales, Area Code
Rows: 4200

ProductTable
---------------
Columns: Product Type, Product, ProductID, Type
Rows: 13

LocationTable
---------------
Columns: AreaCode, State, Market, Market Size
Rows: 156

*/

--Tasks to be performed:

-- 1. Display the number of states present in the LocationTable.

select * from location

select count(distinct state) as NumOfStates from location

---The DISTINCT function in SQL is used to remove duplicate rows from the result
  set of a query.
--It ensures that the returned results contain unique rows only.

--2: How many products are of regular type?
select * from Product
```

```sql
select count(Product) from Product where Type='Regular'

----3.How much spending has been done on marketing of product ID 1?(fact)

select sum(Marketing) as TotalMarketingSpend
from fact
where ProductId =1

--4.What is the minimum sales of a product?

Select min(sales) from fact

--To find the ProductID with the minimum sales, you can use a subquery to first
  determine the minimum sales,
--and then retrieve the corresponding ProductID. Here's how you can write the SQL
  query:

select productID, sales
from fact
where sales = (Select min(sales) from fact)

----5.Display the max Cost of Good Sold (COGS).

select max(COGS) from fact

----6. Display the details of the product where product type is coffee.
select * from product where product_type = 'Coffee'

--7. Display the details where total expenses are greater than 40.

select * from fact

select * from fact where Total_expenses >40

--8. What is the average sales in area code 719?

select avg(sales) from fact where Area_code=719

--To display Area code with average sales :
--Use GROUP BY to aggregate data based on specific columns.

SELECT Area_Code, AVG(Sales)
FROM Fact
WHERE Area_Code = 719
GROUP BY Area_Code;

--Note: You can't use the following query to display the average sales with area
  code:
```

```sql
select area_code, sales
from fact
where Sales=(select avg(sales) from fact where Area_code=719 )

--When calculating averages, you get a summary of a group rather than a value for
  each individual row.
--For example, the average sales won't exactly match any single row's sales
  amount, so you can't directly compare it like you would with the minimum value.

/*
Quick Reference:

MIN with Subquery: Use this to find details related to the smallest value in your
  data.
Example: Find the product with the lowest sales.

AVG with GROUP BY: Use this to calculate the average value for each group and show
   these averages with their group identifiers.
Example: Calculate the average sales for each area code.

*/

--9. Find out the total profit generated by Colorado state.

select area_code,State from location where
state='colorado'

select sum(profit) as TotalProfit from fact
where Area_code in( 303,719,720,970)

--2nd way :

select sum(profit) as Total_Profit
from fact f
inner join location l
on
f.area_code=l.area_code
where state='Colorado'

--10. Display the average inventory for each product ID.

select Productid,avg(inventory) as AvgInventory
from fact
group by productid
order by productid

--11. Display state in a sequential order in a Location Table.

select * from location
```

```
order by state

-----12. Display the average budget of the Product where the average budget margin
   should be greater than 100.

select productid, avg(budget_margin)
from fact
group by productid
having avg(budget_margin)>100
order by productid

/*
GROUP BY is used to group data based on productid so that aggregate functions can
  be applied to each group.
HAVING is used to filter the results of these aggregations.
To filter the results of aggregated data (e.g., only include products where the
  average budget margin is greater than 100).
ORDER BY is used to sort the final results based on one or more columns.
*/



--13. What is the total sales done on date 2010-01-01?

select sum(sales) from fact where Date='2010-01-01'

----14. Display the average total expense of each product ID on an individual
  date.

select Date , Productid, avg(Total_expenses) as AvgTotalExpense
from fact
group by date,productid
order by date,productid

---15. Display the table with the following attributes
--date, productID, product_type, product, sales, profit, state, area_code.

select f.date, p.productid, p.product_type, p.product, f.sales, f.profit, l.state,
   l.Area_Code from fact f
inner join location l
on f.Area_code=l.Area_code
inner join product p
on p.productid=f.productid

--16. Display the rank without any gap to show the sales wise rank.

select Date,Productid,sales,profit,area_code ,
dense_rank() over(order by sales ) as SalesRank
from fact
```

```sql
---17. Find the state wise profit and sales.

Select state ,sum( profit ) as Profit, sum(sales) as Sales from fact f
inner join location l
on f.Area_code = l.Area_code
group by state
order by state

---18. Find the state wise profit and sales along with the productname.

Select l.state , p.product, sum( profit ) as Profit, sum(sales) as Sales from fact
   f
inner join location l
on f.Area_code = l.Area_code
inner join product p
on f.productid=p.productid
group by l.state,p.product
order by l.state,p.product

-----19. If there is an increase in sales of 5%, calculate the increased sales.
--(5%=1+0.05)

select sales ,(sales*1.05) as increased sales
from fact

-----20. Find the maximum profit along with the product ID and producttype.

select p.ProductId, p.Product_type ,max(profit) as profit
from fact f
inner join
product p
on f.productid=p.productid
group by p.productid,p.product_type

----21. Create a stored procedure to fetch the result according to the product
  type
--from Product Table.

create proc ptype(@prod_type varchar(20))
as
begin
select * from product
where product_type=@prod_type
end

ptype @prod_type='coffee'
exec ptype @prod_type='coffee'
execute ptype @prod_type='tea'
```

```
/*
Definition: A stored procedure is a collection of SQL statements stored in the database.
Usage: It can be used to perform tasks like querying, updating, and deleting data.

*/

--22. Write a query by creating a condition in which if the total expenses is less than 60 then it is a profit or else loss.

select Total_expenses,
iif(Total_expenses<60,'profit','loss') as status
from fact

---23. Give the total weekly sales value with the date and product ID details.Use roll-up to pull the data in hierarchical order.

select Datepart(week,date) as WeekNum ,Productid,sum(sales) as WeeklySales
from fact
group by Productid,datepart(week,date) with rollup

--without rollup

select Datepart(week,date) as WeekNum ,Productid,sum(sales) as WeeklySales
from fact
group by Productid,datepart(week,date)

--24. Apply union and intersection operator on the tables which consist of attribute area code.

select Area_code from fact
union
select Area_code from location
order by Area_code

select Area_code from fact
intersect
select Area_code from location
order by Area_code

---25. Create a user-defined function for the product table to fetch a particular
--product type based upon the user's preference.

create function prod(@prod_type varchar(20))
returns table
as
return
select * from product
```

```sql
where product_type = @prod_type

select * from dbo.prod('tea')
select * from dbo.prod('coffee')

/*
* The RETURNS TABLE part of the function signature is declarative. It tells SQL Server what type of data structure (a table in this case) the function will return.
* The RETURN statement inside the function is procedural. It provides the actual query that generates the table to be returned.

* RETURNS: Declares the return type of the function.
* RETURN: Specifies the query that produces the return value.

*/

---26. Change the product type from coffee to tea where product ID is 1 and undo it.

begin transaction
update product
set product_type='tea'
where Productid=1

rollback transaction

begin transaction
update product
set product_type='GreenTea'
where productid in (2,3)

rollback transaction

---27. Display the date, product ID and sales where total expenses are between 100 to 200.

select Date, Productid, sales , Total_expenses from fact
where Total_expenses  between 100 and 200


---28. Delete the records in the Product Table for regular type.

delete product
where type = 'regular'


/*
In SQL, the DELETE statement permanently removes records from a table, and there
```

```
  is no built-in way to "undo" this operation once it has been committed,
especially if you haven't explicitly taken steps to safeguard against accidental
  deletions.

Here are a few approaches to potentially recover the deleted data:

1-Roll back if within a transaction.
2-Restore from backups if available.
3-Set up logging and triggers for future protection.
4-Use point-in-time recovery if supported and configured.
5-Manually re-enter data if no other options are available.

*/

---29. Display the ASCII value of the fifth character from the columnProduct.

select product ,Ascii(substring(product,5,1)) as Charr from product
select * from product

select * from fact
select * from product
select * from location
select * from fact
```