

**SQL DATA**

**EXTRACTION**



# CONTENTS

- **Knowledge of the database we working on**
- **Query Snippets Along with Questions**

# Different Tables in Our Database gdb0041

	date	product_code	customer_code	sold_quantity
►	2017-09-01	A0118150101	70002017	51
	2017-09-01	A0118150101	70002018	77
	2017-09-01	A0118150101	70003181	17
	2017-09-01	A0118150101	70003182	6
	2017-09-01	A0118150101	70006157	5
	2017-09-01	A0118150101	70006158	7
	2017-09-01	A0118150101	70007198	29
	2017-09-01	A0118150101	70007199	34
	2017-09-01	A0118150101	70008169	22
	2017-09-01	A0118150101	70008170	5
	2017-09-01	A0118150101	70011193	10
	2017-09-01	A0118150101	70011194	4
	2017-09-01	A0118150101	70012042	0
	2017-09-01	A0118150101	70012043	0
	2017-09-01	A0118150101	70013125	1
	2017-09-01	A0118150101	70013126	1
	2017-09-01	A0118150101	70016178	1
	2017-09-01	A0118150101	70022085	20
	2017-09-01	A0118150101	70023031	4
	2017-09-01	A0118150101	80001019	10
	2017-09-01	A0118150101	80006154	10
	2017-09-01	A0118150101	80006155	28
	2017-09-01	A0118150101	80007195	80
	2017-09-01	A0118150101	90001021	1
	2017-09-01	A0118150101	90002001	42

”This table captures information about each product sold by every customer during specific months, spanning from fiscal year 2018 to fiscal year 2022.”

fact\_sales\_monthly

	date	fiscal_year	product_code	customer_code	forecast_quantity
►	2017-09-01	2018	A0118150101	70002017	18
	2017-09-01	2018	A0118150101	70002018	11
	2017-09-01	2018	A0118150101	70003181	9
	2017-09-01	2018	A0118150101	70003182	6
	2017-09-01	2018	A0118150101	70006157	5
	2017-09-01	2018	A0118150101	70006158	6
	2017-09-01	2018	A0118150101	70007198	4
	2017-09-01	2018	A0118150101	70007199	7
	2017-09-01	2018	A0118150101	70008169	7
	2017-09-01	2018	A0118150101	70008170	8
	2017-09-01	2018	A0118150101	70011193	5
	2017-09-01	2018	A0118150101	70011194	7
	2017-09-01	2018	A0118150101	70013125	2
	2017-09-01	2018	A0118150101	70013126	2
	2017-09-01	2018	A0118150101	70022085	12
	2017-09-01	2018	A0118150101	70023031	1
	2017-09-01	2018	A0118150101	80001019	7
	2017-09-01	2018	A0118150101	80006154	21
	2017-09-01	2018	A0118150101	80006155	21
	2017-09-01	2018	A0118150101	80007195	62
	2017-09-01	2018	A0118150101	90001021	2
	2017-09-01	2018	A0118150101	90002001	52
	2017-09-01	2018	A0118150101	90002002	47
	2017-09-01	2018	A0118150101	90002003	56
	2017-09-01	2018	A0118150101	90002005	40

”This table consists of forecasted values provided by the supply chain team for each product, outlining the potential demand from each customer for every month, across fiscal years 2018 to 2022.”

fact\_forecast\_monthly

	product_code	fiscal_year	gross_price
►	A0118150101	2018	15.3952
	A0118150101	2019	14.4392
	A0118150101	2020	16.2323
	A0118150101	2021	19.0573
	A0118150102	2018	19.5875
	A0118150102	2019	18.5595
	A0118150102	2020	19.8577
	A0118150102	2021	21.4565
	A0118150103	2018	19.3630
	A0118150103	2019	19.3442
	A0118150103	2020	22.1317
	A0118150103	2021	21.7795
	A0118150103	2022	23.9920
	A0118150104	2018	19.5743
	A0118150104	2019	18.5072
	A0118150104	2020	20.7734
	A0118150104	2021	22.9729
	A0118150104	2022	23.6298

”This table contains the required gross price for each product across the fiscal year from 2018 to 2022.”

fact\_gross\_price



# Different Tables in Our Database gdb0041

	market	fiscal_year	freight_pct	other_cost_pct
►	Australia	2018	0.0188	0.0050
	Australia	2019	0.0304	0.0048
	Australia	2020	0.0254	0.0043
	Australia	2021	0.0254	0.0043
	Australia	2022	0.0254	0.0043
	Austria	2018	0.0272	0.0053
	Austria	2019	0.0292	0.0050
	Austria	2020	0.0294	0.0034
	Austria	2021	0.0294	0.0034
	Austria	2022	0.0294	0.0034
	Banglad...	2018	0.0219	0.0058
	Banglad...	2019	0.0249	0.0053
	Banglad...	2020	0.0258	0.0035
	Banglad...	2021	0.0258	0.0035
	Banglad...	2022	0.0258	0.0035
	Brazil	2018	0.0239	0.0033
	Brazil	2019	0.0193	0.0046
	Brazil	2020	0.0298	0.0046
	Brazil	2021	0.0298	0.0046
	Brazil	2022	0.0298	0.0046
	Canada	2018	0.0264	0.0054
	Canada	2019	0.0217	0.0030
	Canada	2020	0.0273	0.0055
	Canada	2021	0.0273	0.0055
	Canada	2022	0.0273	0.0055

”This table contains the freight cost and associated cost percentages for each country in every fiscal year.”  
fact\_freight\_cost

	product_code	cost_year	manufacturing_cost
►	A0118150101	2018	4.6190
	A0118150101	2019	4.2033
	A0118150101	2020	5.0207
	A0118150101	2021	5.5172
	A0118150102	2018	5.6036
	A0118150102	2019	5.3235
	A0118150102	2020	5.7180
	A0118150102	2021	6.2835
	A0118150103	2018	5.9469
	A0118150103	2019	5.5306
	A0118150103	2020	6.3264
	A0118150103	2021	6.5900
	A0118150103	2022	7.1831
	A0118150104	2018	5.8958
	A0118150104	2019	5.4242
	A0118150104	2020	6.4789
	A0118150104	2021	6.8199
	A0118150104	2022	7.3655
	A0219150201	2019	5.7250
	A0219150201	2020	6.4858
	A0219150201	2021	7.0498
	A0219150201	2022	7.5433
	A0219150202	2019	6.2366
	A0219150202	2020	7.0590
	A0219150202	2021	7.2031

”””This table contains the manufacturing costs associated with each product for every fiscal year.”  
fact\_manufacturing\_cost

	customer_code	fiscal_year	pre_invoice_discount_pct
►	70002017	2018	0.0824
	70002017	2019	0.0777
	70002017	2020	0.0735
	70002017	2021	0.0703
	70002017	2022	0.1057
	70002018	2018	0.2956
	70002018	2019	0.2577
	70002018	2020	0.2255
	70002018	2021	0.2061
	70002018	2022	0.2931
	70003181	2018	0.0536
	70003181	2019	0.0546
	70003181	2020	0.0531
	70003181	2021	0.0974
	70003181	2022	0.0727
	70003182	2018	0.2378
	70003182	2019	0.2023
	70003182	2020	0.1823
	70003182	2021	0.2065
	70003182	2022	0.2147
	70004069	2018	0.0547
	70004069	2019	0.0997
	70004069	2020	0.0576
	70004069	2021	0.1068
	70004069	2022	0.0895

”This table contains the pre-invoice deduction percentages associated with each customer for every fiscal year.”  
fact\_pre\_invoice\_deductions



# Different Tables in Our Database gdb0041

	customer_code	product_code	date	discounts_pct	other_deductions_pct
▶	70002017	A0118150101	2017-09-01	0.2660	0.0719
	70002017	A0118150101	2017-10-01	0.3090	0.0976
	70002017	A0118150101	2017-11-01	0.3313	0.0752
	70002017	A0118150101	2018-01-01	0.2958	0.0720
	70002017	A0118150101	2018-02-01	0.3208	0.0793
	70002017	A0118150101	2018-03-01	0.2635	0.1007
	70002017	A0118150101	2018-05-01	0.2231	0.0820
	70002017	A0118150101	2018-06-01	0.3020	0.0791
	70002017	A0118150101	2018-07-01	0.3123	0.0929
	70002017	A0118150101	2018-09-01	0.1530	0.1288
	70002017	A0118150101	2018-10-01	0.1363	0.1542
	70002017	A0118150101	2018-11-01	0.1521	0.1641
	70002017	A0118150101	2019-01-01	0.1762	0.1386
	70002017	A0118150101	2019-02-01	0.1844	0.1635
	70002017	A0118150101	2019-03-01	0.1628	0.1413
	70002017	A0118150101	2019-05-01	0.1553	0.1555
	70002017	A0118150101	2019-06-01	0.1595	0.1198
	70002017	A0118150101	2019-07-01	0.1760	0.1365
	70002017	A0118150101	2019-09-01	0.2981	0.1101
	70002017	A0118150101	2019-10-01	0.2354	0.0858
	70002017	A0118150101	2019-11-01	0.3056	0.1091
	70002017	A0118150101	2020-01-01	0.3240	0.0890
	70002017	A0118150101	2020-02-01	0.3220	0.1187
	70002017	A0118150101	2020-03-01	0.3026	0.1109
	70002017	A0118150101	2020-05-01	0.2792	0.1098
	70002017	A0118150101	2020-06-01	0.2256	0.0824

”This table contains the post-invoice discounts and other discounts for each customer and each product on a monthly basis, spanning from fiscal year 2018 to 2022.”

fact\_post\_invoice\_deductions

	customer_code	customer	platform	channel	market	sub_zone	region
▶	70002017	Atliq Exclusive	Brick & Mortar	Direct	India	India	APAC
	70002018	Atliq e Store	E-Commerce	Direct	India	India	APAC
	70003181	Atliq Exclusive	Brick & Mortar	Direct	Indonesia	ROA	APAC
	70003182	Atliq e Store	E-Commerce	Direct	Indonesia	ROA	APAC
	70004069	Atliq Exclusive	Brick & Mortar	Direct	Japan	ROA	APAC
	70004070	Atliq e Store	E-Commerce	Direct	Japan	ROA	APAC
	70005163	Atliq e Store	E-Commerce	Direct	Pakistan	ROA	APAC
	70006157	Atliq Exclusive	Brick & Mortar	Direct	Philippines	ROA	APAC
	70006158	Atliq e Store	E-Commerce	Direct	Philippines	ROA	APAC
	70007198	Atliq Exclusive	Brick & Mortar	Direct	South Korea	ROA	APAC
	70007199	Atliq e Store	E-Commerce	Direct	South Korea	ROA	APAC
	70008169	Atliq Exclusive	Brick & Mortar	Direct	Australia	ANZ	APAC
	70008170	Atliq e Store	E-Commerce	Direct	Australia	ANZ	APAC
	70009133	Atliq Exclusive	Brick & Mortar	Direct	Newzealand	ANZ	APAC
	70009134	Atliq e Store	E-Commerce	Direct	Newzealand	ANZ	APAC
	70010047	Atliq Exclusive	Brick & Mortar	Direct	Bangladesh	ROA	APAC
	70010048	Atliq e Store	E-Commerce	Direct	Bangladesh	ROA	APAC
	70011193	Atliq Exclusive	Brick & Mortar	Direct	France	SE	EU
	70011194	Atliq e Store	E-Commerce	Direct	France	SE	EU
	70012042	Atliq Exclusive	Brick & Mortar	Direct	Germany	NE	EU
	70012043	Atliq e Store	E-Commerce	Direct	Germany	NE	EU
	70013125	Atliq Exclusive	Brick & Mortar	Direct	Italy	SE	EU
	70013126	Atliq e Store	E-Commerce	Direct	Italy	SE	EU
	70014142	Atliq Exclusive	Brick & Mortar	Direct	Netherlands	NE	EU
	70014143	Atliq e Store	E-Commerce	Direct	Netherlands	NE	EU
	70015154	Atliq Exclusive	Brick & Mortar	Direct	Norway	NE	EU
	70015155	Atliq e Store	E-Commerce	Direct	Norway	NE	EU

”This table contains various details for each customer based on their customer code, including the customer’s name, the platform channel market to which they belong, as well as their subzone and region.”

dim\_customer

	product_code	division	segment	category	product	variant
▶	A0118150101	P & A	Peripherals	Internal HDD	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Standard
	A0118150102	P & A	Peripherals	Internal HDD	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Plus
	A0118150103	P & A	Peripherals	Internal HDD	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Premium
	A0118150104	P & A	Peripherals	Internal HDD	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache	Premium Plus
	A0219150201	P & A	Peripherals	Internal HDD	AQ WereWolf NAS Internal Hard Drive HDD – 8.89 cm	Standard
	A0219150202	P & A	Peripherals	Internal HDD	AQ WereWolf NAS Internal Hard Drive HDD – 8.89 cm	Plus
	A0220150203	P & A	Peripherals	Internal HDD	AQ WereWolf NAS Internal Hard Drive HDD – 8.89 cm	Premium
	A0320150301	P & A	Peripherals	Internal HDD	AQ Zion Saga	Standard
	A0321150302	P & A	Peripherals	Internal HDD	AQ Zion Saga	Plus
	A0321150303	P & A	Peripherals	Internal HDD	AQ Zion Saga	Premium
	A0418150101	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Standard 1
	A0418150102	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Standard 2
	A0418150103	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Standard 3
	A0418150104	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Plus 1
	A0418150105	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Plus 2
	A0418150106	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Plus 3
	A0418150107	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Premium 1
	A0418150108	P & A	Peripherals	Graphic Card	AQ Mforce Gen X	Premium 2
	A0519150201	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Standard 1
	A0519150202	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Standard 2
	A0519150203	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Standard 3
	A0519150204	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Plus 1
	A0519150205	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Plus 2
	A0519150206	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Plus 3
	A0519150207	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Premium 1
	A0519150208	P & A	Peripherals	Graphic Card	AQ Mforce Gen Y	Premium 2

”This table contains various information about products based on their product code, including the product name, division, segment category, and variant name.”

dim\_product

**"QUERY SNIPPETS  
ALONG WITH  
QUESTIONS"**



## Question 1

**Generate a report of individual product sales (aggregated on a monthly basis at the product code level) for Croma India customer for FY-2021**

**"The report requires the following fields: month, product name, variant, sold quantity, gross price per item, and total gross price."**

```
select s.date,s.product_code,p.product,p.variant,s.sold_quantity,gp.gross_price,gp.gross_price*s.sold_quantity as total_gross
from fact_sales_monthly as s
join dim_product as p on s.product_code=p.product_code
join fact_gross_price as gp on gp.product_code=s.product_code and gp.fiscal_year=fiscal_year(s.date)
where fiscal_year(date)=2021 and customer_code=(select customer_code from dim_customer where customer="croma")
order by date asc ;
```

**"We join the dim\_product table with the fact\_sales\_monthly table using product\_code to retrieve the product name and variant. Similarly, we join the fact\_gross\_price table with the fact\_sales\_monthly table by product\_code and fiscal year to obtain the gross price of the product. Next, we apply a WHERE clause to specify both the fiscal year and the customer using the customer\_code. Since we do not have the customer\_code for Croma, we utilize a subquery to return the customer code based on the provided store name from the dim\_customer table.**

**gross price of a product=sold quantity\*gross price of that product  
Additionally, you will notice an undefined function known as fiscal\_year, which calculates the fiscal year based on the date provided to the function."**

```
1 CREATE DEFINER='root'@'localhost' FUNCTION `fiscal_year`(  
2   calender_date date) RETURNS int  
3   DETERMINISTIC  
4 BEGIN  
5   declare fiscal_year int;  
6   set fiscal_year=year(date_add(calender_date ,interval 4 month));  
7   RETURN fiscal_year;  
8 END
```

## Question 2

**Monthly Gross Sales Report for  
Croma India**  
Report Fields:

**Month:** Month of the sale  
**Total Gross Sales Amount:** Total  
revenue generated from sales to  
Croma India in the month

```
select s.date,round(sum(g.gross_price*sold_quantity),2) as total_gross from fact_sales_monthly s
join fact_gross_price g
on s.product_code=g.product_code and fiscal_year(s.date)=g.fiscal_year
join dim_product p on p.product_code=s.product_code
where s.customer_code=(select customer_code from dim_customer where customer="croma")
GROUP BY
    s.date
order by s.date asc;
```

"We join the fact\_gross\_price table with the fact\_sales\_monthly table by product\_code and fiscal\_year to obtain the gross price of each product. Next, we apply a WHERE clause to the customer using the customer\_code. Since we do not have the customer\_code for Croma, we utilize a subquery to return the customer code based on the provided store name from the dim\_customer table. Additionally, we group the data by date, which allows us to aggregate the gross prices for each individual date using the SUM function."

**gross price of a product=sold quantity\*gross price of that product**

```
1 CREATE DEFINER='root'@'localhost' FUNCTION `fiscal_year`(  
2     calender_date date) RETURNS int  
3     DETERMINISTIC  
4 BEGIN  
5     declare fiscal_year int;  
6     set fiscal_year=year(date_add(calender_date ,interval 4 month));  
7     RETURN fiscal_year;  
8 END
```



## Question 3

### Yearly Gross Sales Report for Croma India

#### Report Fields:

**Fiscal Year:** Year of the sale  
**Total Gross Sales Amount:** Total revenue generated from sales to Croma India in the fiscal year

```
select fiscal_year(date) as fiscal_year,round(sum(g.gross_price*sold_quantity),2) as total_gross from fact_sales_monthly s
join fact_gross_price g
on s.product_code=g.product_code and fiscal_year(s.date)=g.fiscal_year
join dim_product p on p.product_code=s.product_code
where s.customer_code=(select customer_code from dim_customer where customer="croma")
group by fiscal_year(s.date)
order by fiscal_year asc;
```

"We join the fact\_gross\_price table with the fact\_sales\_monthly table by product\_code and fiscal\_year to obtain the gross price of each product. Next, we apply a WHERE clause to the customer using the customer\_code. Since we do not have the customer\_code for Croma, we utilize a subquery to return the customer code based on the provided store name from the dim\_customer table. Additionally, we group the data by fiscal year of the date, which allows us to aggregate the gross prices for each individual fiscal year using the SUM function."

**gross price of a product=sold quantity\*gross price of that product**

```
1 CREATE DEFINER='root'@'localhost' FUNCTION `fiscal_year`(  
2   calender_date date) RETURNS int  
3   DETERMINISTIC  
4 BEGIN  
5   declare fiscal_year int;  
6   set fiscal_year=year(date_add(calender_date ,interval 4 month));  
7   RETURN fiscal_year;  
8 END
```

## Question 4

**"Write a SQL query to find the top ten customers who have the highest net sales market share, including their market percentages and total net sales in millions."**

```
with hresult as(select s.date,s.product_code,s.customer_code,s.sold_quantity,s.fiscal_year,g.gross_price,
g.gross_price*sold_quantity as tot_gross,
pre.pre_invoice_discount_pct*gross_price*sold_quantity as tot_preinvoice from fact_sales_monthly s
join fact_gross_price g
on g.product_code=s.product_code
and g.fiscal_year=s.fiscal_year
join fact_pre_invoice_deductions pre
on pre.customer_code=s.customer_code
and s.fiscal_year=pre.fiscal_year
),
net_invoice_value as (
select hr.date,hr.customer_code,hr.product_code,hr.sold_quantity,hr.fiscal_year,hr.gross_price,hr.tot_gross,hr.tot_preinvoice,tot_gross-tot_preinvoice as net_invoice from hresult hr),
post_invoice_value as( select inv.product_code,inv.customer_code,inv.sold_quantity,inv.fiscal_year,inv.gross_price,inv.tot_gross,inv.tot_preinvoice,inv.net_invoice,
(inv.net_invoice*pos.discounts_pct+inv.net_invoice*other_deductions_pct) as tot_post from net_invoice_value inv
join fact_post_invoice_deductions pos
on pos.customer_code=inv.customer_code and pos.product_code=inv.product_code and inv.date=pos.date
),
top_customer as (select c.customer,round(sum((pos.net_invoice-pos.tot_post)/1000000),2) as net_sales_Million from post_invoice_value pos
join dim_customer c on c.customer_code=pos.customer_code
where fiscal_year=2021
group by customer,fiscal_year
order by net_sales_Million desc
),
chart1 as (select *,round((net_sales_Million)*100/sum(net_sales_Million) over(),2) as netsales_percent from top_customer)

select * from chart1
limit 10;
```



**In the first Common Table Expression (CTE) named hresult, we calculate the total gross price of products sold on a specific date by customers, using the fact\_sales\_monthly table. We also compute the pre-invoice amount for the corresponding sales. The fact\_gross\_price is joined through the fiscal year and product code, while the fact\_pre\_invoice\_deductions table is joined using the customer code and fiscal year. Since our fact table does not contain a fiscal year, we add a calculated column to the fact\_sales\_monthly table to determine the fiscal year based on the sale date, allowing us to accurately identify when the sales transactions occurred. This calculation of the fiscal year is performed prior to the execution of the hresult CTE.**

**In the second Common Table Expression (CTE) net\_invoice\_value, we calculate the net\_invoice by subtracting the total pre-invoice amount, calculated in the hresult CTE, from the total gross price.**

**"In the third CTE, post\_invoice\_value, we join the fact\_post\_invoice\_deductions table using the date, customer\_code, and product\_code to compute the total post-invoice amount. Furthermore, we join this CTE with the dim\_customer table to obtain the customer name based on the customer\_code. We then group the results by customer and fiscal year, incorporating a WHERE clause to specify the fiscal year. Finally, we apply the SUM aggregate function to calculate the total, which is derived from the net invoice amount minus the post-invoice amount"**

**"Additionally, we form another CTE by joining the third CTE with the dim\_customer table to retrieve the customer name associated with the customer\_code. In this step, we also group the results by customer and fiscal year, using a WHERE clause to specify the fiscal year, while again employing the SUM aggregate function to calculate the net sales, which is based on the net invoice minus the postinvoice amount."**



## Question 5

**"Write a SQL query to find the top 5 markets who have the highest net sales in Millions"**

```
with hresult as(select s.date,s.product_code,s.customer_code,s.sold_quantity,s.fiscal_year,g.gross_price,
g.gross_price*sold_quantity as tot_gross,
pre.pre_invoice_discount_pct*gross_price*sold_quantity as tot_preinvoice from fact_sales_monthly s
join fact_gross_price g
on g.product_code=s.product_code
and g.fiscal_year=s.fiscal_year
join fact_pre_invoice_deductions pre
on pre.customer_code=s.customer_code
and s.fiscal_year=pre.fiscal_year
join fact_post_invoice_deductions pos
on pos.customer_code=s.customer_code
and pos.product_code=s.product_code
and s.date=pos.date
),
net_invoice_value as (
select hr.date,hr.customer_code,hr.product_code,hr.sold_quantity,hr.fiscal_year,hr.gross_price,hr.tot_gross,hr.tot_preinvoice,
tot_gross-tot_preinvoice as net_invoice from hresult hr),
post_invoice_value as( select inv.customer_code,inv.sold_quantity,inv.fiscal_year,inv.gross_price,inv.tot_gross,inv.tot_preinvoice,inv.net_invoice,
(inv.net_invoice*pos.discounts_pct+inv.net_invoice*other_deductions_pct) as tot_post from net_invoice_value inv
join fact_post_invoice_deductions pos
on pos.customer_code=inv.customer_code and pos.product_code=inv.product_code and inv.date=pos.date
),
top_market as (select pos.fiscal_year,c.market,round(sum((pos.net_invoice-pos.tot_post)/1000000),2) as net_salesM from post_invoice_value pos
join dim_customer c on c.customer_code=pos.customer_code
where fiscal_year=2021
group by market,fiscal_year
order by net_salesM desc limit 5
)
select * from top_market;
```

**In the first Common Table Expression (CTE) named hresult, we calculate the total gross price of products sold on a specific date by customers, using the fact\_sales\_monthly table. We also compute the pre-invoice amount for the corresponding sales. The fact\_gross\_price is joined through the fiscal year and product code, while the fact\_pre\_invoice\_deductions table is joined using the customer code and fiscal year. Since our fact table does not contain a fiscal year, we add a calculated column to the fact\_sales\_monthly table to determine the fiscal year based on the sale date, allowing us to accurately identify when the sales transactions occurred. This calculation of the fiscal year is performed prior to the execution of the hresult CTE.**

**In the second Common Table Expression (CTE) net\_invoice\_value, we calculate the net\_invoice by subtracting the total pre-invoice amount, calculated in the hresult CTE, from the total gross price.**

**"In the third CTE, post\_invoice\_value, we join the fact\_post\_invoice\_deductions table using the date, customer\_code, and product\_code to compute the total post-invoice amount. Furthermore, we join this CTE with the dim\_customer table to obtain the customer name based on the customer\_code. We then group the results by customer and fiscal year, incorporating a WHERE clause to specify the fiscal year. Finally, we apply the SUM aggregate function to calculate the total, which is derived from the net invoice amount minus the post-invoice amount"**



**"Additionally, we form another CTE by joining the third CTE with the dim\_customer table to retrieve the Market associated with the customer\_code. In this step, we also group the results by market and fiscal year, using a WHERE clause to specify the fiscal year, while again employing the SUM aggregate function to calculate the netsales, which is based on the net invoice minus the post invoice amount."**

## Question 6

**"Write a SQL query to find the top 5 sold products along with thier division and segment who have the highest net sales in Millions"**

```
with hresult as(select s.date,s.product_code,s.customer_code,s.sold_quantity,s.fiscal_year,g.gross_price,
g.gross_price*sold_quantity as tot_gross,
pre.pre_invoice_discount_pct*gross_price*sold_quantity as tot_preinvoice from fact_sales_monthly s
join fact_gross_price g
on g.product_code=s.product_code
and g.fiscal_year=s.fiscal_year
join fact_pre_invoice_deductions pre
on pre.customer_code=s.customer_code
and s.fiscal_year=pre.fiscal_year
join fact_post_invoice_deductions pos
on pos.customer_code=s.customer_code
and pos.product_code=s.product_code
and s.date=pos.date
),
net_invoice_value as (
select hr.date,hr.customer_code,hr.product_code,hr.sold_quantity,hr.fiscal_year,hr.gross_price,hr.tot_gross,
hr.tot_preinvoice,tot_gross-tot_preinvoice as net_invoice from hresult hr),
post_invoice_value as( select inv.product_code,inv.customer_code,inv.sold_quantity,inv.fiscal_year,inv.gross_price,inv.tot_gross,
inv.tot_preinvoice,inv.net_invoice,
(inv.net_invoice*pos.discounts_pct+inv.net_invoice*other_deductions_pct) as tot_post from net_invoice_value inv
join fact_post_invoice_deductions pos
on pos.customer_code=inv.customer_code and pos.product_code=inv.product_code and inv.date=pos.date
),
top_product as (select p.division,p.segment,p.product,round(sum((pos.net_invoice-pos.tot_post)/1000000),2) as net_salesM from post_invoice_value pos
join dim_product p on p.product_code=pos.product_code
where fiscal_year=2021
group by division,segment,product
order by net_salesM desc limit 5
)
select * from top_product;
```

**In the first Common Table Expression (CTE) named hresult, we calculate the total gross price of products sold on a specific date by customers, using the fact\_sales\_monthly table. We also compute the pre-invoice amount for the corresponding sales. The fact\_gross\_price is joined through the fiscal year and product code, while the fact\_pre\_invoice\_deductions table is joined using the customer code and fiscal year. Since our fact table does not contain a fiscal year, we add a calculated column to the fact\_sales\_monthly table to determine the fiscal year based on the sale date, allowing us to accurately identify when the sales transactions occurred. This calculation of the fiscal year is performed prior to the execution of the hresult CTE.**

**In the second Common Table Expression (CTE) net\_invoice\_value, we calculate the net\_invoice by subtracting the total pre-invoice amount, calculated in the hresult CTE, from the total gross price.**

**"In the third CTE, post\_invoice\_value, we join the fact\_post\_invoice\_deductions table using the date, customer\_code, and product\_code to compute the total post-invoice amount. Furthermore, we join this CTE with the dim\_customer table to obtain the customer name based on the customer\_code. We then group the results by customer and fiscal year, incorporating a WHERE clause to specify the fiscal year. Finally, we apply the SUM aggregate function to calculate the total, which is derived from the net invoice amount minus the post-invoice amount"**



**Furthermore, we join the dim\_product table to the latest CTE using product\_code to retrieve the division and segment names of the products. We also use the SUM aggregate function to aggregate the net sales amount and group the results by division, segment, and product to identify the top five products."**

## Question 7

**"Write a SQL Query to find of the top three Markets in a region by their gross sales in Millions and Display them along with their position in fiscal year 2021"**

```
• with cte as(select m.market,m.region,round(sum(s.sold_quantity*g.gross_price)/1000000,2) as total_gross from fact_sales_monthly s
join fact_gross_price g on
g.fiscal_year=s.fiscal_year and g.product_code=s.product_code
join dim_customer m
on m.customer_code=s.customer_code
where s.fiscal_year=2021
group by market,region
),
cte2 as(select *,dense_rank() over(partition by region order by total_gross desc) as Position from cte)
select * from cte2
where
Position<=3
```

**"As you can see, in the first CTE, the fact\_sales\_monthly table is joined with the fact\_gross\_price table by fiscal\_year and product\_code to determine the gross price of the products sold. Additionally, the fact\_sales\_monthly table is joined with the dim\_customer table to aggregate the gross price in terms of market and region by grouping them, while applying a WHERE clause to specify the fiscal year.**

**Subsequently, another CTE is utilized to calculate the dense rank of different countries within a region based on gross price. We then display the market region along with the gross price and their position, filtering for those ranked third or lower. This allows us to identify the top three countries in a region based on their gross price."**

## Question 8

```
with cte as(select p.division,p.segment,p.category,p.product,sum(sold_quantity) as total from fact_sales_monthly s
join dim_product p on s.product_code=p.product_code
where fiscal_year=2021
group by p.division,p.segment,p.category,p.product),
cte2 as(select *,(dense_rank() over(partition by division order by total desc)) as position from cte)
select * from cte2
where position<=3
```

**"Write an SQL query to find out the Top 3 products sold in a Division by their quantity in Fiscal Year 2021.Print those products along with division,segment category and position"**

**"As you can see in the first CTE, we join the fact\_sales\_monthly table with the dim\_product table on product\_code to retrieve the product name, segment, category, and division. We then group the results by division, segment, category, and product. Even though products may share the same division, category, and segment, we include them in the GROUP BY clause since they cannot be aggregated directly. We utilize the SUM aggregate function to aggregate the sold\_quantity.**

**In the next CTE, we apply the DENSE\_RANK function to rank the products within each division based on their sold quantity. We then display the top three products according to their rank, allowing us to identify the leading products in each division."**



## Question 9

"Requirement for a report that will show the forecast accuracy for all customers in 2021 fiscal year. The report should include the following fields:

**Customer Code, Name, and Market**  
**Total Sold Quantity**  
**Total Forecast Quantity**  
**Net Error**  
**Absolute Error**  
**Forecast Accuracy %"**

```
select count(*) from fact_sales_monthly;
select count(*) from fact_forecast_monthly;
select count(*) from fact_sales_monthly s join fact_forecast_monthly f
on s.date=f.date and s.product_code=f.product_code and s.customer_code=f.customer_code;
create table fact_act_est as(
select s.date,s.product_code,s.customer_code,s.sold_quantity,f.forecast_quantity
  from fact_sales_monthly s
left join fact_forecast_monthly f
on s.date=f.date and f.product_code=s.product_code and f.customer_code=s.customer_code
union
select f.date,f.product_code,f.customer_code,s.sold_quantity,f.forecast_quantity
  from fact_sales_monthly s
right join fact_forecast_monthly f
on s.date=f.date and f.product_code=s.product_code and f.customer_code=s.customer_code
);
```

"The fact\_sales\_monthly table consists of 1,425,706 records, while the fact\_forecast\_monthly table contains 1,885,841 records. The union of these two tables results in 1,390,837 records. From this, we identified that some records in fact\_sales\_monthly do not have corresponding forecast quantities, and some records in fact\_forecast\_monthly lack sales quantities.

To address this, we decided to perform a full outer join by unioning the results of left and right joins between the fact\_sales\_monthly and fact\_forecast\_monthly tables. In the left join, fact\_sales\_monthly serves as the left table and fact\_forecast\_monthly as the right. All matching forecast quantities are retrieved, and where no match is found, NULL is returned. These tables are joined using the date, product\_code, and customer\_code.

Similarly, in the right join, fact\_sales\_monthly is the left table, and fact\_forecast\_monthly is the right. Here, we retrieve the matching sales quantities, returning NULL for any missing matches. The results from both the left and right joins are then unioned to form the final table, fact\_act\_est."

```
alter table fact_act_est
add column fiscal_year int
generated always as (year(date_add(date ,interval 4 month))) stored;
update fact_act_est
set sold_quantity=0
where sold_quantity is null;
update fact_act_est
set forecast_quantity=0
where forecast_quantity is null;
select * from fact_act_est;
```

"Next, we add a calculated column, fiscal\_year, to the newly created table. Since the company's fiscal year starts in September, we adjust the dates by adding 4 months. After adjusting the date, we extract the year from the updated date to determine the correct fiscal\_year."

"Next, we update the sold\_quantity and forecast\_quantity columns, setting any NULL values to 0. This is a common practice in the industry when handling missing values, ensuring that analyses can proceed without gaps in the data."

```

DELIMITER $$
CREATE TRIGGER before_insert_sales
BEFORE INSERT ON fact_sales_monthly
FOR EACH ROW
BEGIN
INSERT INTO fact_act_est (date, product_code, customer_code, sold_quantity)
VALUES (NEW.date, NEW.product_code, NEW.customer_code, NEW.sold_quantity)
ON DUPLICATE KEY UPDATE
sold_quantity = VALUES(sold_quantity);
END$$
DELIMITER ;
DELIMITER $$
CREATE TRIGGER before_insert_forecast
BEFORE INSERT ON fact_forecast_monthly
FOR EACH ROW
BEGIN
INSERT INTO fact_act_est (date, product_code, customer_code, forecast_quantity)
VALUES (NEW.date, NEW.product_code, NEW.customer_code, NEW.forecast_quantity)
ON DUPLICATE KEY UPDATE
forecast_quantity = VALUES(forecast_quantity);
END$$
DELIMITER ;

```

"Now, we create triggers such that whenever a new entry is made into either the fact\_sales\_monthly or fact\_forecast\_monthly table, the trigger is automatically activated. This ensures that the corresponding record is also inserted into the fact\_act\_est table, keeping it updated in real time."



```
show triggers;
```

```
with cte1 as(select fiscal_year,customer_code,sum(sold_quantity) as sold_quantity,sum(forecast_quantity) as forecast_quantity,  
sum((cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))) as net_error,  
sum((cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))*100/sum(forecast_quantity) as net_error_pct,  
sum(abs(cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))) as abs_error,  
sum(abs(cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))*100/sum(forecast_quantity) as abs_error_pct from fact_act_est  
  
group by fiscal_year,customer_code  
  
)
```

"In the first CTE (cte1), we calculate the net\_error, net\_error\_percentage, abs\_error, and abs\_error\_percentage using the fact\_act\_est table.

Net\_error is the difference between forecast\_quantity and sold\_quantity.

Abs\_error represents the real error, calculated by taking the absolute difference between forecast\_quantity and sold\_quantity for each product sold per day. For example, if the net\_error is 1, -1, 1, -1 over four days, the total net\_error is 0, but the abs\_error becomes 4 as it takes the absolute values (1, 1, 1, 1).

Net\_error\_percentage is the net difference between forecast and sold quantities, multiplied by 100 and divided by the forecast quantity.

Abs\_error\_percentage is similar but uses the absolute difference instead.

Casting forecast\_quantity and sold\_quantity as signed integers ensures correct handling of negative values in these calculations."

```
select cte1.fiscal_year,c.customer,c.customer_code,c.market,cte1.sold_quantity,cte1.forecast_quantity,net_error,net_error_pct,abs_error,abs_error_pct,
case
when abs_error_pct is null then null
when abs_error_pct is not null then (100-abs_error_pct)
end as forecast_accuracy
from cte1 join dim_customer c
on c.customer_code=cte1.customer_code
where fiscal_year=2021
order by forecast_accuracy desc;
```

"Since we have already grouped the values by fiscal\_year and customer\_code, we now join the CTE with the dim\_customer table on customer\_code to retrieve the customer name and market. Next, we check if the abs\_error\_percentage or net\_error\_percentage is null. If not, we calculate the forecast accuracy by subtracting the abs\_error\_percentage from 100."

## Question 10

"Requirement for a report that will show all customers whose forecast accuracy 2021 is lesser than 2020 following fields:

Customer\_code, customer, market, forecast\_accuracy\_2021 and forecast\_accuracy\_2020

```
with cte1 as(select fiscal_year,customer_code,sum(sold_quantity) as sold_quantity,sum(forecast_quantity) as forecast_quantity,
sum((cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))) as net_error,
sum((cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))*100/sum(forecast_quantity) as net_error_pct,
sum(abs(cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))) as abs_error,
sum(abs(cast(forecast_quantity as signed int)-cast(sold_quantity as signed int))*100/sum(forecast_quantity) as abs_error_pct from fact_act_est

group by fiscal_year,customer_code

),

cte2 as(select cte1.fiscal_year,c.customer,c.customer_code,c.market,cte1.sold_quantity,cte1.forecast_quantity,net_error,net_error_pct,abs_error,abs_error_pct,
case
when abs_error_pct is null then null
when abs_error_pct is not null then (100-abs_error_pct)
end as forecast_accuracy
from cte1 join dim_customer c
on c.customer_code=cte1.customer_code
where fiscal_year=2021
order by forecast_accuracy desc),

cte3 as (select cte1.fiscal_year,c.customer,c.customer_code,c.market,cte1.sold_quantity,cte1.forecast_quantity,net_error,net_error_pct,abs_error,abs_error_pct,
case
when abs_error_pct is null then null
when abs_error_pct is not null then (100-abs_error_pct)
end as forecast_accuracy
from cte1 join dim_customer c
on c.customer_code=cte1.customer_code
where fiscal_year=2020
order by forecast_accuracy desc)
select c2.customer_code,c2.customer,c2.market,c3.forecast_accuracy as forecast_accuracy_2020,c2.forecast_accuracy as forecast_accuracy_2021 from cte2 c2 join cte3 c3
on c2.customer_code=c3.customer_code
where c2.forecast accuracy<c3.forecast accuracy;
```



"In the first CTE (cte1), we calculate the net\_error, net\_error\_percentage, abs\_error, and abs\_error\_percentage using the fact\_act\_est table.

Net\_error is the difference between forecast\_quantity and sold\_quantity.

Abs\_error represents the real error, calculated by taking the absolute difference between forecast\_quantity and sold\_quantity for each product sold per day. For example, if the net\_error is 1, -1, 1, -1 over four days, the total net\_error is 0, but the abs\_error becomes 4 as it takes the absolute values (1, 1, 1, 1).

Net\_error\_percentage is the net difference between forecast and sold quantities, multiplied by 100 and divided by the forecast quantity.

Abs\_error\_percentage is similar but uses the absolute difference instead.

Casting forecast\_quantity and sold\_quantity as signed integers ensures correct handling of negative values in these calculations."

"Since we have already grouped the values by fiscal\_year and customer\_code, we now join the CTE with the dim\_customer table on customer\_code to retrieve the customer name and market to form another cte. Next, we check if the abs\_error\_percentage or net\_error\_percentage is null. If not, we calculate the forecast accuracy by subtracting the abs\_error\_percentage from 100. For fiscal\_year 2021 by using a where clause"

"Next, we create a third CTE that includes the values for the fiscal year 2020."

"We then join cte2 with cte3 on customer\_code to compare forecast accuracy for 2020 and 2021. We display the forecast accuracy for both years, along with the customer\_code, customer name, and market, where the forecast accuracy for 2020 is greater than that of 2021."