Task #1: Extract the Customer Code for a specific customer (CROMA)operating in India from 'dim_customer' table.

```
SELECT * FROM dim_customer WHERE customer LIKE "%croma%" AND market =
"India";
```

Output Snippet #1:

customer_code	customer	platform	channel	market	sub_zone	region
90002002	Croma	Brick & Mortar	Retailer	India	India	APAC
NULL	NULL	NULL	NULL	NULL	NULL	NULL

USER DEFINED FUNCTIONS :

Task #2: Create a user-defined function `get_fiscal_year` to get corresponding fiscal year (which begins in September for ths company) , by passing the calender date .

```
CREATE FUNCTION `get_fiscal_year`(calendar_date DATE)
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE fiscal_year INT;
SET fiscal_year = YEAR(DATE_ADD(calendar_date, INTERVAL 4 MONTH));
RETURN fiscal_year;
END
```

Using this function, get all the sales transactions from 'fact_sales_monthly' table for that customer(Croma: 90002002) for fiscal_year: 2021

```
SELECT *
FROM fact_sales_monthly
WHERE customer_code=90002002 AND get_fiscal_year(DATE)=2021
ORDER BY DATE ASC
LIMIT 100000;
```

Output Snippet #2: Fiscal Year 2021 starts in September 2020 for this company, so we get all output beginning from date: 2020-09-01.

date	fiscal_year	product_code	customer_code	sold_quantity
2020-09-01	2021	A0118150101	90002002	202
2020-09-01	2021	A0118150102	90002002	162
2020-09-01	2021	A0118150103	90002002	193
2020-09-01	2021	A0118150104	90002002	146
2020-09-01	2021	A0219150201	90002002	149
2020-09-01	2021	A0219150202	90002002	107
2020-09-01	2021	A0220150203	90002002	123
2020-09-01	2021	A0320150301	90002002	146
2020-09-01	2021	A0321150302	90002002	236

Task #3: Create another user-defined function `get_fiscal_quarter` to generate quarter-wise sales data for Croma.

```
CREATE FUNCTION `get_fiscal_quarter`(calendar_date DATE)
RETURNS CHAR(2)
DETERMINISTIC
BEGIN
DECLARE m TINYINT;
DECLARE qtr CHAR(2);
SET m = MONTH(calender_date);
CASE
        WHEN m in (9,10,11) THEN SET qtr = "Q1";
    WHEN m in (12,1,2) THEN SET qtr = "Q2"; WHEN m
    in (3,4,5) THEN SET qtr = "Q3"; WHEN m in (6,7,8)
    THEN SET qtr = "Q4";
END CASE ;
RETURN qtr;
END
```

Use this function to get 'sold_quantity' data for Croma for Fiscal Year 2021 for Quarter 4.

```
SELECT date, sold_quantity
FROM fact_sales_monthly
WHERE customer_code = 90002002 AND get_fiscal_year(date) = 2021 AND
get_fiscal_quarter(date) = "Q4"
ORDER BY DATE ASC;
```

Output Snippet #3:

date	sold_quantity
2021-06-01	205
2021-06-01	78
2021-06-01	48
2021-06-01	126
2021-06-01	40
2021-06-01	102
2021-06-01	31
2021-06-01	91
2021-06-01	70
2021-06-01	145

JOINING TABLES

Task #4: Get Gross Sales Report: Monthly Product-level sales data for Croma, by joining two tables – 'fact_sales_monthly' and 'dim_product'.

```
SELECT s.date, p.product_code, p.product, p.variant, s.sold_quantity
FROM fact_sales_monthly AS s
JOIN dim_product AS p
ON s.product_code = p.product_code
WHERE customer_code = 90002002 AND get_fiscal_year(date) = 2021
ORDER BY DATE ASC;
```

Output Snippet #4:

date	product_code	product	variant	sold_quantity
2020-12-01	A0118150101	AQ Dracula HDD…	Standard	113
2020-12-01	A0118150102	AQ Dracula HDD…	Plus	172
2020-12-01	A0118150103	AQ Dracula HDD…	Premium	383
2020-12-01	A0118150104	AQ Dracula HDD…	Premium Plus	199
2020-12-01	A0219150201	AQ WereWolf NA…	Standard	362
2020-12-01	A0219150202	AQ WereWolf NA…	Plus	409
2020-12-01	A0220150203	AQ WereWolf NA…	Premium	121
2020-12-01	A0320150301	AQ Zion Saga	Standard	364
2020-12-01	A0321150302	AQ Zion Saga	Plus	339

Task #5: Get Gross Price and Gross Price Total amounts from another table, 'fact_gross_price using another Join.

```
SELECT s.date, p.product_code, p.product, p.variant, s.sold_quantity,
        ROUND (g.gross_price,3) AS gross_price,
      ROUND (g.gross_price*s.sold_quantity,2) AS gross_price_total
FROM fact_sales_monthly AS s
JOIN dim_product AS p
ON s.product_code = p.product_code
JOIN fact_gross_price AS g
ON g.product_code = s.product_code AND g.fiscal_year =
get_fiscal_year(s.date)
WHERE customer_code = 90002002 AND get_fiscal_year(date) = 2021
ORDER BY DATE ASC;
```

Output Snippet #5:

date	product_code	product	variant	sold_quantity	gross_price	gross_price_total
2020-09-01	A0118150101	AQ Dracula HDD…	Standard	202	19.057	3849.57
2020-09-01	A0118150102	AQ Dracula HDD…	Plus	162	21.457	3475.95
2020-09-01	A0118150103	AQ Dracula HDD…	Premium	193	21.780	4203.44
2020-09-01	A0118150104	AQ Dracula HDD…	Premium Plus	146	22.973	3354.04
2020-09-01	A0219150201	AQ WereWolf NA…	Standard	149	23.699	3531.11
2020-09-01	A0219150202	AQ WereWolf NA…	Plus	107	24.731	2646.24
2020-09-01	A0220150203	AQ WereWolf NA…	Premium	123	23.615	2904.69
2020-09-01	A0320150301	AQ Zion Saga	Standard	146	23.722	3463.46
2020-09-01	A0321150302	AQ Zion Saga	Plus	236	27.103	6396.24

Task #6: Get a Gross Sales Report for Croma by joining 'fact_sales_monthly' and 'fact_gross_price' tables.

```
SELECT s.date,
        ROUND(SUM(g.gross_price*s.sold_quantity),2) AS gross_sales_amount
FROM fact_sales_monthly AS s
JOIN fact_gross_price AS g
ON g.product_code = s.product_code AND g.fiscal_year =
get_fiscal_year(s.date)
WHERE customer_code = 90002002
GROUP BY s.date
ORDER BY s.date ASC;
```

Output Snippet #6:

date	gross_sales_amount
2017-09-01	122407.56
2017-10-01	162687.57
2017-12-01	245673.80
2018-01-01	127574.74
2018-02-01	144799.52
2018-04-01	130643.90
2018-05-01	139165.10
2018-06-01	125735.38
2018-08-01	125409.88

Task #7: Get Total Yearly Sales Report for Croma using FiscalYear

```
SELECT get_fiscal_year(s.date) AS fiscal_year,
      ROUND(SUM(g.gross_price*s.sold_quantity),2) AS yearly_sales
FROM fact_sales_monthly AS s
JOIN fact_gross_price AS g
ON g.product_code = s.product_code AND g.fiscal_year = get_fiscal_year(s.date)
WHERE customer_code = 90002002
GROUP BY fiscal_year
ORDER BY fiscal_year ASC;
```

Output Snippet #7:

fiscal_year	yearly_sales
2022	44638198.92
2021	23216512.22
2020	6502181.91
2019	3555079.02
2018	1324097.44

STORED PROCEDURE

Task #8: Create a stored procedure to get Monthly Gross Sales Report for any Customer using their customer code.

```
CREATE PROCEDURE `get_monthly_gross_sales_for_customer`(c_code INT)
BEGIN
   SELECT s.date, SUM(ROUND(s.sold_quantity*g.gross_price,2)) AS monthly_sales
   FROM fact_sales_monthly s
   JOIN fact_gross_price g
   ON g.fiscal_year = get_fiscal_year(s.date) AND g.product_code = s.product_code
   WHERE customer_code = c_code
   GROUP BY s.date;
END
```

Then call this stored procedure to get Gross Monthly Sales for Croma using its customer code.

```
CALL get_monthly_gross_sales_for_customer(90002002);
```

Output Snippet #8.1:

date	monthly_sales
2017-09-01	122407.57
2017-10-01	162687.56
2017-12-01	245673.84
2018-01-01	127574.73
2018-02-01	144799.54
2018-04-01	130643.92
2018-05-01	139165.06
2018-06-01	125735.36
2018-08-01	125409.90

Sometimes a customer has two or more codes, so the stored procedure needs to accommodate this need. For example, let's say Amazon has two product codes so we create a stored procedure for it:

```
CREATE PROCEDURE `get_monthly_gross_sales_for_amazon`(
      in_customer_codes TEXT
) BEGIN
SELECT s.date,
       SUM(ROUND(s.sold_quantity*g.gross_price,2)) AS monthly_sales
FROM fact_sales_monthly s
JOIN fact_gross_price g
ON g.fiscal_year=get_fiscal_year(s.date) AND g.product_code=s.product_code
WHERE FIND_IN_SET(s.customer_code, in_customer_codes) > 0
GROUP BY s.date
ORDER BY s.date DESC;
END
```

We can then call the stored procedure using the two product codes for Amazon:

```
CALL get_monthly_gross_sales_for_amazon('90002016, 90002008');
```

Output Snippet #8.2:

date	monthly_sales
2021-12-01	19849946.98
2021-11-01	19289758.70
2021-10-01	14634931.44
2021-08-01	2316472.95
2021-07-01	2380093.58
2021-06-01	2406323.75
2021-04-01	2276252.96
2021-03-01	2300637.79
2021-02-01	2337095.49

Task #9: Create a stored procedure to give Market Badges to customers based on their sales turnover.

First we need to create a join between 'fact_sales_monthly' and 'dim_customer' table to get total market-wise (country-wise) sales quantities.

```
SELECT c.market, SUM(sold_quantity) AS total_qty
FROM fact_sales_monthly s
JOIN dim_customer c ON s.customer_code = c.customer_code
WHERE get_fiscal_year(s.date) = 2021
GROUP BY c.market;
```

Output Snippet #9:

market	total_qty
India	13751429
Indonesia	1434929
Japan	529487
Pakistan	454393
Philiphines	2422641
South Korea	3947794
Australia	1782354
Newzealand	835190
Bangladesh	575892
France	2047367

Using this output we can develop a stored procedure such that if sold quantity for market is greater than 5 million for the chosen year, that market is 'Gold',

```
CREATE PROCEDURE `get_market_badge`(
        IN in_market VARCHAR(45),
    IN in_fiscal_year YEAR,
    OUT out_badge VARCHAR(45)
)
BEGIN
    DECLARE qty INT DEFAULT 0;
     -- Set default market as india
        IF in_market = "" THEN
            SET in_market = "India";
        END IF;
-- Retrieve total sold quantity for given market and given fiscal_year
      SELECT sum(s.sold_quantity) INTO qty FROM fact_sales_monthly s
      JOIN dim_customer c
      ON c.customer_code = s.customer_code
      WHERE
          get_fiscal_year(s.date) = in_fiscal_year and
          c.market = in_market
      GROUP BY c.market;
-- Determine market badge ('GOLD' vs 'SILVER')
      IF qty > 5000000 THEN
      SET out_badge = "Gold";
      ELSE SET out_badge = "Silver";
  END IF ;
END
```

Now upon calling this stored procedure, for market = India and year = 2021, we get the output as: Gold. This is correct, as India's Sales for this year is '13751429'.

```
SET @out_badge = '0'; CALL get_market_badge('India', 2021, @out_badge); SELECT
@out_badge;
```

@out_badge
Gold

market	total_qty
India	13751429
Indonesia	1434929
Japan	529487
Pakistan	454393
Philiphines	2422641
South Korea	3947794
Australia	1782354
Newzealand	835190
Bangladesh	575892
France	2047367

Task #10: Add a 'fiscal_year' column to 'fact_sales_monthly' table, and use it to join with 'pre invoice deductions' table to get 'pre_invoice_discount_pct'.

```sql
SELECT
	s.date,
	s.customer_code,
	s.product_code,
	p.product,
	p.variant,
	s.sold_quantity,
	g.gross_price AS gross_price_per_item,
	ROUND(s.sold_quantity*g.gross_price,2)  AS  gross_price_total,
	pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
JOIN dim_product p
ON s.product_code=p.product_code
JOIN fact_gross_price g
ON g.fiscal_year=s.fiscal_year AND g.product_code=s.product_code
JOIN fact_pre_invoice_deductions AS pre
ON pre.customer_code = s.customer_code AND pre.fiscal_year=s.fiscal_year
WHERE
	 s.fiscal_year=2021
LIMIT 1500000;
```

Output Snippet #10:

date	customer_code	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct
2020-09-01	70002017	A6120110205	AQ HOME Allin1...	Plus 2	4	834.9812	3339.92	0.0703
2020-09-01	70002018	A6120110205	AQ HOME Allin1...	Plus 2	1	834.9812	834.98	0.2061
2020-09-01	70003181	A6120110205	AQ HOME Allin1...	Plus 2	2	834.9812	1669.96	0.0974
2020-09-01	70003182	A6120110205	AQ HOME Allin1...	Plus 2	1	834.9812	834.98	0.2065
2020-09-01	70006157	A6120110205	AQ HOME Allin1...	Plus 2	1	834.9812	834.98	0.0858
2020-09-01	70006158	A6120110205	AQ HOME Allin1...	Plus 2	1	834.9812	834.98	0.2450
2020-09-01	70007198	A6120110205	AQ HOME Allin1...	Plus 2	1	834.9812	834.98	0.0736
2020-09-01	70007199	A6120110205	AQ HOME Allin1...	Plus 2	2	834.9812	1669.96	0.2105

DATABASE VIEWS

Task #11: Create a view named `sales_preinv_discount` to store and access all current data captured for future reuse.

```sql
CREATE VIEW `sales_preinv_discount` AS
SELECT
	s.date,
	s.fiscal_year,
	 s.customer_code,
	c.customer,
	c.market,
	s.product_code,
	p.product,
	p.variant,
	s.sold_quantity,
	g.gross_price AS gross_price_per_item,
	ROUND(s.sold_quantity  *  g.gross_price,2)  AS  gross_price_total,
	pre.pre_invoice_discount_pct
FROM fact_sales_monthly s
```

```
JOIN dim_customer c
ON s.customer_code = c.customer_code
JOIN dim_product p
ON s.product_code = p.product_code
JOIN fact_gross_price g
ON g.fiscal_year = s.fiscal_year AND g.product_code = s.product_code
JOIN fact_pre_invoice_deductions AS pre
ON pre.customer_code = s.customer_code AND pre.fiscal_year = s.fiscal_year
```

Use this view to generate 'net_invoice_sales' data:

```
SELECT *,
    ((1-pre_invoice_discount_pct)* gross_price_total) AS net_invoice_sales
FROM sales_preinv_discount
```

Output Snippet #11:

date	fiscal_year	customer_code	customer	market	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct	net_invoice_sales
2017-09-01	2018	70002017	Atliq Exclusive	India	A0118150101	AQ Dracula HDD...	Standard	51	15.3952	785.16	0.0824	720.462816
2017-09-01	2018	70002018	Atliq e Store	India	A0118150101	AQ Dracula HDD...	Standard	77	15.3952	1185.43	0.2956	835.016892
2017-09-01	2018	70003181	Atliq Exclusive	Indonesia	A0118150101	AQ Dracula HDD...	Standard	17	15.3952	261.72	0.0536	247.691808
2017-09-01	2018	70003182	Atliq e Store	Indonesia	A0118150101	AQ Dracula HDD...	Standard	6	15.3952	92.37	0.2378	70.404414
2017-09-01	2018	70006157	Atliq Exclusive	Philiphines	A0118150101	AQ Dracula HDD...	Standard	5	15.3952	76.98	0.1057	68.843214
2017-09-01	2018	70006158	Atliq e Store	Philiphines	A0118150101	AQ Dracula HDD...	Standard	7	15.3952	107.77	0.1875	87.563125
2017-09-01	2018	70007198	Atliq Exclusive	South Ko...	A0118150101	AQ Dracula HDD...	Standard	29	15.3952	446.46	0.0700	415.207800
2017-09-01	2018	70007199	Atliq e Store	South Ko...	A0118150101	AQ Dracula HDD...	Standard	34	15.3952	523.44	0.2551	389.910456
2017-09-01	2018	70008169	Atliq Exclusive	Australia	A0118150101	AQ Dracula HDD...	Standard	22	15.3952	338.69	0.0953	306.412843
2017-09-01	2018	70008170	Atliq e Store	Australia	A0118150101	AQ Dracula HDD...	Standard	5	15.3952	76.98	0.1896	62.384592

Task #12: Create a view `sales_postinv_discount` to capture 'post_invoice_discount_pct':

```
CREATE VIEW `sales_postinv_discount` AS
SELECT
        s.date,
        s.fiscal_year,
        s.customer_code,
        s.market,
        s.customer,
        s.product_code,
        s.product,
        s.variant,
        s.sold_quantity,
        s.gross_price_total,
        s.pre_invoice_discount_pct,
        (1-s.pre_invoice_discount_pct) * s.gross_price_total) AS net_invoice_sales,
        (po.discounts_pct+po.other_deductions_pct) AS post_invoice_discount_pct
FROM sales_preinv_discount s
JOIN fact_post_invoice_deductions po
ON po.customer_code = s.customer_code AND po.product_code = s.product_code
AND po.date = s.date;
```

Using this view, create a report for 'net_sales':

```
SELECT *,
    net_invoice_sales*(1-post_invoice_discount_pct) AS net_sales
FROM sales_postinv_discount;
```

Output Snippet #12:

date	fiscal_year	customer_code	market	customer	product_code	product	variant	sold_quantity	gross_price_total	pre_invoice_discount_pct	net_invoice_sales	post_invoice_discount_pct	net_sales
2017-09-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	4	61.58	0.2803	44.319126	0.3905	27.0125072970
2017-11-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	16	246.32	0.2803	177.276504	0.4139	103.9017589944
2017-12-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	4	61.58	0.2803	44.319126	0.3295	29.7159739830
2018-01-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	6	92.37	0.2803	66.478689	0.3244	44.9130022884
2018-03-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	9	138.56	0.2803	99.721632	0.3766	62.1664653888
2018-04-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	6	92.37	0.2803	66.478689	0.3615	42.4466429265
2018-05-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	7	107.77	0.2803	77.562069	0.3173	52.9516245063
2018-07-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	10	153.95	0.2803	110.797815	0.3501	72.0074999685
2018-08-01	2018	90027207	Brazil	Amazon	A0118150101	AQ Dracula HD...	Standard	6	92.37	0.2803	66.478689	0.3740	41.6156593140

Now save 'net_sales' as a view for future use:

```
CREATE VIEW `net_sales` AS
SELECT *,
  net_invoice_sales*(1-post_invoice_discount_pct) AS net_sales
FROM sales_postinv_discount;
```

Task #13: Get Top 5 Market (Countries) by Net Sales in Fiscal Year 2021

```
SELECT
    market,
    ROUND(SUM(net_sales)/1000000, 2) AS net_sales_mln
FROM net_sales
WHERE fiscal_year=2021
GROUP BY market
ORDER BY net_sales_mln DESC
LIMIT 5;
```

Output Snippet #13:

market	net_sales_mln
India	210.67
USA	132.05
South Korea	64.01
Canada	45.89
United Kingdom	44.73

Task #14: Create a stored procedure to get Top N Markets by Net Sales for a chosen year.

```
CREATE PROCEDURE `get_top_n_market_by_net_sales`(
        in_fiscal_year YEAR,
        in_top_n INT
)
BEGIN
    SELECT
            market,
            ROUND(SUM(net_sales)/1000000,2) AS net_sales_mln
    FROM gdb0041.net_sales
WHERE fiscal_year = in_fiscal_year
GROUP BY market
ORDER BY net_sales_mln DESC
LIMIT in_top_n ;
END
```

Use this stored procedure to see Top 5 markets in 2020:

```
CALL get_top_n_markets_by_net_sales(2020, 5);
```

Output Snippet #14:

market	net_sales_mln
India	64.73
USA	46.35
South Korea	22.38
Philiphines	17.45
Canada	15.87

Task #15: Create a stored procedure to get Top N Customers by Net Sales for a chosen year.

```
CREATE PROCEDURE `get_top_n_customers_by_net_sales`(
    in_market VARCHAR(45),
    in_fiscal_year INT,
    in_top_n INT
)
BEGIN
    SELECT
            customer,
            ROUND(SUM(net_sales)/1000000,2) AS net_sales_mln
    FROM net_sales s
    WHERE s.fiscal_year = in_fiscal_year AND s.market = in_market
    GROUP BY customer
    ORDER BY net_sales_mln DESC
    LIMIT in_top_n;
END
```

Call this stored procedure to see Top 5 customers in Japan in 2020:

```
CALL get_top_n_customers_by_net_sales('Japan', 2020, 5);
```

Output Snippet #15:

customer	net_sales_mln
Amazon	0.76
Atliq e Store	0.32
Atliq Exclusive	0.23
All-Out	0.21
Surface Stores	0.19

Task #16: Create a stored procedure to get the Top N Products by Net Sales for a chosen year.

```
CREATE PROCEDURE `get_top_n_products_by_net_sales`(
    in_fiscal_year int,
    in_top_n int
)
BEGIN
    SELECT
            product,
            ROUND(SUM(net_sales)/1000000,2) AS net_sales_mln
    FROM net_sales
    WHERE fiscal_year=in_fiscal_year
    GROUP BY product
    ORDER BY net_sales_mln DESC
    LIMIT in_top_n;
END
```

Call this stored procedure to see data for Top 5 Products in 2021:

```
CALL get_top_n_products_by_net_sales(2021, 5);
```

product	net_sales_mln
AQ BZ Allin1	33.75
AQ Qwerty	27.84
AQ Trigger	26.95
AQ Gen Y	23.58
AQ Maxima	22.32

COMMON TABLE EXPRESSION & WINDOW FUNCTION

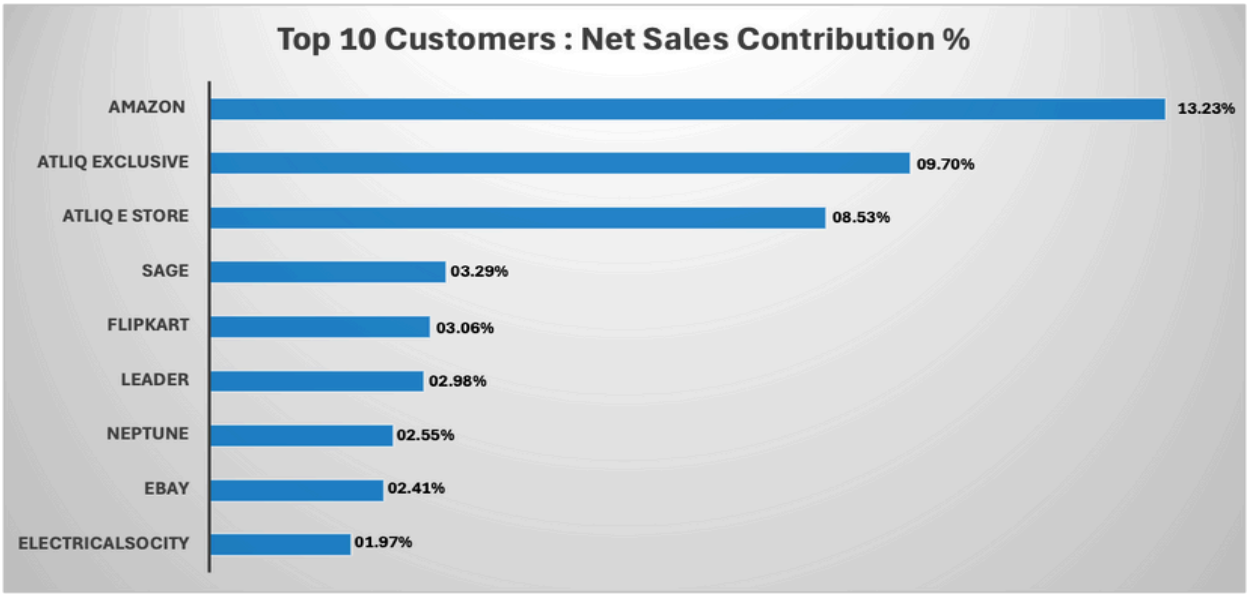
Task #17: Find out customer-wise Net Sales percentage contribution using the stored procedure for Top N Customers by modifying the query to create a Common table Expression (CTE) on which we can create a Window Function .

```
WITH cte1 AS (
    SELECT
            customer,
           ROUND(SUM(net_sales)/1000000,2) AS net_sales_mln
    FROM net_sales s
      WHERE s.fiscal_year = 2021
      GROUP BY customer)
SELECT
    *,
    net_sales_mln*100/SUM(net_sales_mln) OVER() AS pct_net_sales
FROM cte1
ORDER BY net_sales_mln DESC;
```

Output #17:

customer	net_sales_mln	pct_net_sales
Atliq Exclusive	79.92	9.700206
Atliq e Store	70.31	8.533803
Sage	27.07	3.285593
Flipkart	25.25	3.064692
Leader	24.52	2.976089
Neptune	21.01	2.550067
Ebay	19.88	2.412914

Visual Chart of Top 10 Customers by Net Sales Contribution in % (graphed after extracting the data in Excel)



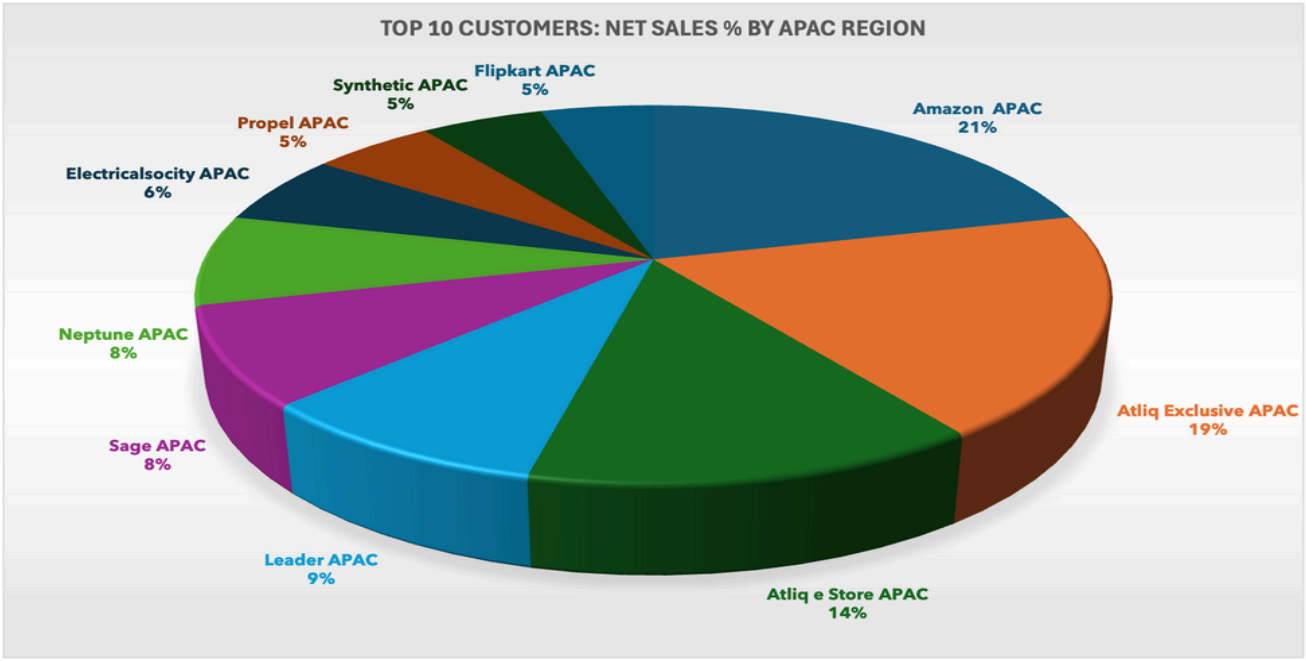
Task #18: Find Customer-wise Net Sales distribution per region for Fiscal Year 2021.

```sql
WITH cte1 AS (
    SELECT
        c.customer,
        c.region,
        ROUND(SUM(net_sales)/1000000,2) AS net_sales_mln
    FROM net_sales n
    JOIN dim_customer c
    ON n.customer_code=c.customer_code
    WHERE fiscal_year=2021
    GROUP BY c.customer, c.region)
SELECT
    *,
    net_sales_mln*100/SUM(net_sales_mln) OVER (PARTITION BY region) AS pct_share_region
FROM cte1
ORDER BY region, pct_share_region DESC;
```

Output Snippet #18:

customer	region	net_sales_mln	pct_share_region
Amazon	APAC	57.41	12.988688
Atliq Exclusive	APAC	51.58	11.669683
Atliq e Store	APAC	36.97	8.364253
Leader	APAC	24.52	5.547511
Sage	APAC	22.85	5.169683
Neptune	APAC	21.01	4.753394
Electricalsocity	APAC	16.25	3.676471
Propel	APAC	14.14	3.199095
Synthetic	APAC	14.14	3.199095
Flipkart	APAC	12.96	2.932127

Pie Chart representing Top 10 Customers for APAC region: (graphed after extracting the data in Excel)



Task #19: Find Top 3 Products from each Division by total quantity sold in a given year using DENSE_RANK.

```
WITH cte1 AS (
    SELECT
            p.division,
            p.product,
            SUM(sold_quantity) AS total_qty
    FROM fact_sales_monthly s JOIN dim_product p ON p.product_code=s.product_code
    WHERE fiscal_year=2021
    GROUP BY p.product,p.division),
    cte2 AS (
            SELECT *,
            DENSE_RANK() OVER(PARTITION BY division
            ORDER BY total_qty DESC) AS drnk
            FROM cte1)
SELECT * FROM cte2 WHERE drnk<=3;
```

Output Snippet #19.1:

division	product	total_qty	drnk
N & S	AQ Pen Drive DRC	2034569	1
N & S	AQ Digit SSD	1240149	2
N & S	AQ Clx1	1238683	3
P & A	AQ Gamers Ms	2477098	1
P & A	AQ Maxima Ms	2461991	2
P & A	AQ Master wireless x1 Ms	2448784	3
PC	AQ Digit	135092	1
PC	AQ Gen Y	135031	2
PC	AQ Elite	134431	3

Create a stored procedure for the above query and call it to see Top 2 Products in each Division for 2020:

```
CALL get_top_n_products_per_division_by_qty_sold(2020, 2);
```

Output Snippet #19.2:

division	product	total_qty	drnk
N & S	AQ Clx1	935128	1
N & S	AQ Neuer SSD	924264	2
P & A	AQ Master wired x1 Ms	1578253	1
P & A	AQ Gamers Ms	1566445	2
PC	AQ Digit	68862	1
PC	AQ Elite	67841	2

Task #20.0 : Create a view `gross_sales` to capture 'total_gross_price':

```
CREATE VIEW `gross_sales` AS
        SELECT
            s.date,
            s.fiscal_year,
            s.product_code,
            s.sold_quantity,
            p.product,
            p.variant,
            s.customer_code,
            c.region,
            g.gross_price,
            ROUND((g.gross_price * s.sold_quantity),2) AS gross_price_total
        From fact_sales_monthly s
        JOIN dim_product p
        ON p.product_code = s.product_code
        JOIN dim_customer c
        ON c.customer_code = s.customer_code
        JOIN fact_gross_price g
        ON g.product_code = s.product_code AND g.fiscal_year = s.fiscal_year;
```

Task #20.1: Retrieve the Top 2 Markets in every Region by their gross sales amount in Fiscal Year 2021.

```
WITH cte1 AS(
    SELECT
        gs.market,
         c.region,
         ROUND(SUM(gs.gross_price_total)/1000000,2) AS gross_sales_mln
    FROM gross_sales AS gs
    JOIN dim_customer AS c ON gs.customer_code = c.customer_code
    WHERE fiscal_year=2021
    GROUP BY gs.market ,c.region
    ORDER BY c.region),
    cte2 AS(
        SELECT
            *,
             DENSE_RANK() OVER(PARTITION BY region ORDER BY gross_sales_mln DESC) AS rnk
         FROM cte1)
SELECT * FROM cte2 WHERE rnk<=2;
```

Output Snippet #20:

market	region	gross_sales_mln	rnk
India	APAC	455.05	1
South Korea	APAC	131.86	2
United Kingdom	EU	78.11	1
France	EU	67.62	2
Mexico	LATAM	2.30	1
Brazil	LATAM	2.14	2
USA	NA	264.46	1
Canada	NA	89.78	2

Task #21: Create a Helper Table for Forecast Accuracy Reports with Fiscal Year Calculations .

```
CREATE TABLE fact_actual_estimate
(
   SELECT
        s.date,
        s.product_code,
        s.customer_code,
        s.fiscal_year,
        s.sold_quantity,
        f.forecast_quantity
   FROM gdb0041.fact_sales_monthly s
   LEFT JOIN fact_forecast_monthly f
   USING(date,product_code, customer_code)
   UNION
   SELECT
        f.date,
        f.product_code,
        f.customer_code,
        f.fiscal_year,
        s.sold_quantity,
        f.forecast_quantity
   FROM gdb0041.fact_forecast_monthly f
   LEFT JOIN fact_sales_monthly s
   USING(date,product_code, customer_code)
);
```

Output Snippet #21.0:

date	product_code	customer_code	fiscal_year	sold_quantity	forecast_quantity
2017-09-01	A0118150101	70002017	2018	51	18
2017-09-01	A0118150101	70002018	2018	77	11
2017-09-01	A0118150101	70003181	2018	17	9
2017-09-01	A0118150101	70003182	2018	6	6
2017-09-01	A0118150101	70006157	2018	5	5
2017-09-01	A0118150101	70006158	2018	7	6
2017-09-01	A0118150101	70007198	2018	29	4
2017-09-01	A0118150101	70007199	2018	34	7
2017-09-01	A0118150101	70008169	2018	22	7
2017-09-01	A0118150101	70008170	2018	5	8
2017-09-01	A0118150101	70011193	2018	10	5
2017-09-01	A0118150101	70011194	2018	4	7
2017-09-01	A0118150101	70012042	2018	0	NULL
2017-09-01	A0118150101	70012043	2018	0	NULL

We also need to update our sold_quantity and forecast_quantity column where there are null values

```
UPDATE fact_actual_estimate
SET sold_quantity = 0
WHERE sold_quantity IS NULL;

UPDATE fact_actual_estimate
SET forecast_quantity = 0
WHERE forecast_quantity IS NULL;
```

Use this to view 'fact_actual_estimate' data .

```sql
SELECT * FROM fact_actual_estimate;
```

Output Snippet #21.1:

Task #22: Calculate net_error_pct and abs_net_error_pct from fact_actual_estimate

```sql
SELECT
        f.*,
        SUM(forecast_quantity - sold_quantity) AS net_error,
        SUM(forecast_quantity - sold_quantity) * 100 / SUM(forecast_quantity) AS  net_error_pct,
        SUM(ABS(forecast_quantity - sold_quantity)) AS abs_net_error,
        SUM(ABS(forecast_quantity - sold_quantity)) * 100 / SUM(forecast_quantity) AS
        abs_net_error_pct
 FROM gdb0041.fact_actual_estimate f
 WHERE f.fiscal_year = 2021
 GROUP BY f.customer_code, f.date, f.product_code
 ORDER BY abs_net_error_pct DESC;
```

Output Snippet #22:

date	fiscal_year	product_code	customer_code	sold_quantity	forecast_quantity	net_error	net_error_pct	abs_net_error	abs_net_error_pct
2020-11-01	2021	A3320150503	70019204	56	2	-54	-2700.0000	54	2700.0000
2021-08-01	2021	A3018150201	70014143	84	3	-81	-2700.0000	81	2700.0000
2021-07-01	2021	A3220150402	90012040	126	5	-121	-2420.0000	121	2420.0000
2021-04-01	2021	A3521150705	90014139	99	4	-95	-2375.0000	95	2375.0000
2020-11-01	2021	A3421150604	90014141	147	6	-141	-2350.0000	141	2350.0000
2021-04-01	2021	A3420150602	90014135	73	3	-70	-2333.3333	70	2333.3333
2021-04-01	2021	A3421150606	90014139	93	4	-89	-2225.0000	89	2225.0000
2020-09-01	2021	A3019150204	90014135	92	4	-88	-2200.0000	88	2200.0000
2021-04-01	2021	A3120150306	70014142	69	3	-66	-2200.0000	66	2200.0000
2020-09-01	2021	A3119150301	90014138	89	4	-85	-2125.0000	85	2125.0000

Note :For calculating forecast accuracy we will use abs_net_error_pct not the net_error_pct because net_error_pct has negative values that will cancel out the error .

Task #23: Create Stored procedure for the given fiscal_year to get forecast accuracy

```sql
CREATE PROCEDURE `forecast_accuracy`(
in_fiscal_year YEAR
)
BEGIN
    WITH forecast_err_table as

    (SELECT
        f.customer_code ,
        SUM(f.sold_quantity) AS total_sold_qty,
        SUM(f.forecast_quantity)AS total_forecast_qty,
        SUM(forecast_quantity - sold_quantity) AS net_error,
        SUM(forecast_quantity - sold_quantity) * 100 / SUM(forecast_quantity) AS net_error_pct,
        SUM(ABS(forecast_quantity - sold_quantity)) AS abs_net_error,
        SUM(ABS(forecast_quantity - sold_quantity)) * 100 / SUM(forecast_quantity) AS
        FROM fact_actual_estimate f
        WHERE f.fiscal_year = in_fiscal_year
        GROUP BY f.customer_code)
    SELECT e.*,c.customer,c.market,
    if (abs_net_error_pct > 100 ,0, (100-abs_net_error_pct)) as forecast_accuracy
    FROM forecast_err_table e
    JOIN dim_customer c
    USING(customer_code)
   ORDER BY forecast_accuracy DESC;
END
```

Call this stored procedure to see data for fiscal_year 2021 to get forecast_accuracy

```
CALL get_forecast_accuracy(2021);
```

Output Snippet #23:

customer_code	total_sold_qty	total_forecast_qty	net_error	net_error_pct	abs_net_error	abs_net_error_pct	customer	market	forecast_accuracy
90013120	109547	133532	23985	17.9620	70467	52.7716	Coolblue	Italy	47.2284
70010048	119439	142010	22571	15.8940	75711	53.3139	Atliq e Store	Bangladesh	46.6861
90023027	236189	279962	43773	15.6353	149303	53.3297	Costco	Canada	46.6703
90023026	228988	273492	44504	16.2725	146948	53.7303	Relief	Canada	46.2697
90017051	86823	118067	31244	26.4629	63568	53.8406	Forward Stores	Portugal	46.1594
90017058	86860	110195	23335	21.1761	59473	53.9707	Mbit	Portugal	46.0293
90023028	239081	283323	44242	15.6154	153058	54.0224	walmart	Canada	45.9776
90023024	246397	287233	40836	14.2170	155610	54.1755	Sage	Canada	45.8245
90013124	110898	136116	25218	18.5268	73826	54.2376	Amazon	Italy	45.7624
90015146	147152	210507	63355	30.0964	114189	54.2448	Mbit	Norway	45.7552
90017054	84371	114698	30327	26.4407	62483	54.4761	Flawless Stores	Portugal	45.5239

"Task 24: Comparative Analysis of Forecast Accuracy — Identifying Customers with Declining Performance from 2020 to 2021.

For this we need to create temporary table for forecast_accuracy for fiscal year 2020 and fiscal year 2021 and then perform join to get the desired result .

Table 1: Forecast Accuracy for fiscal year 2021

```
CREATE TEMPORARY TABLE forecast_accuracy_2021
WITH forecast_err_table AS (
SELECT
    f.customer_code ,
    SUM(f.sold_quantity) AS total_sold_qty,
    SUM(f.forecast_quantity)AS total_forecast_qty,
    SUM(forecast_quantity - sold_quantity) AS net_error,
    SUM(forecast_quantity - sold_quantity) * 100 / SUM(forecast_quantity) AS  net_error_pct,
    SUM(ABS(forecast_quantity - sold_quantity)) AS abs_net_error,
    SUM(ABS(forecast_quantity - sold_quantity)) * 100 / SUM(forecast_quantity) AS abs_net_error_pct,
    FROM fact_actual_estimate f
    WHERE f.fiscal_year = 2021
    GROUP BY f.customer_code)
    SELECT e.*,c.customer,c.market,
    IF (abs_net_error_pct > 100 ,0, (100-abs_net_error_pct)) AS forecast_accuracy
    FROM forecast_err_table e
    JOIN dim_customer c
    USING(customer_code)
    ORDER BY forecast_accuracy DESC;
```

Table 2: Forecast Accuracy for fiscal year 2020

```
CREATE TEMPORARY TABLE forecast_accuracy_2020
WITH forecast_err_table AS (
SELECT
    f.customer_code ,
    SUM(f.sold_quantity) AS total_sold_qty,
    SUM(f.forecast_quantity)AS total_forecast_qty,
    SUM(forecast_quantity - sold_quantity) AS net_error,
    SUM(forecast_quantity - sold_quantity) * 100 / SUM(forecast_quantity)  AS net_error_pct,
```

```
    SUM(ABS(forecast_quantity - sold_quantity)) AS abs_net_error,
    SUM(ABS(forecast_quantity - sold_quantity)) * 100 / SUM(forecast_quantity) AS abs_net_error_pct,
    FROM fact_actual_estimate f
    WHERE f.fiscal_year = 2020
    GROUP BY f.customer_code)
    SELECT e.*,c.customer,c.market,
    IF (abs_net_error_pct > 100 ,0, (100-abs_net_error_pct)) AS forecast_accuracy
    FROM forecast_err_table e JOIN dim_customer c
    USING(customer_code)
    ORDER BY forecast_accuracy DESC;
```

Joining Table 1 and Table 2

```
SELECT
    f_2020.customer_code,
    f_2020.customer,
    f_2020.market,
    f_2020.forecast_accuracy AS forecast_acc_2020,
    f_2021.forecast_accuracy AS forecast_acc_2021
FROM forecast_accuracy_2020 f_2020
JOIN forecast_accuracy_2021 f_2021
ON f_2020.customer_code = f_2021.customer_code
WHERE f_2021.forecast_accuracy < f_2020.forecast_accuracy
ORDER BY forecast_acc_2020 DESC;
```

Output Snippet #23:

customer_code	customer	market	forecast_acc_2020	forecast_acc_2021
70006158	Atliq e Store	Philiphines	42.6505	24.4904
70008170	Atliq e Store	Australia	40.9573	38.7404
90005161	Zone	Pakistan	40.0813	37.0962
90014140	Radio Popular	Netherlands	38.5260	0.0000
90008166	Sound	Australia	38.5111	36.7911
70014143	Atliq e Store	Netherlands	38.3174	0.0000
90004062	Flawless Stores	Japan	38.2162	32.5617
90014137	Media Markt	Netherlands	37.8548	0.0000
90014138	Mbit	Netherlands	37.8277	0.0000
70004069	Atliq Exclusive	Japan	37.6207	32.0943