SQL-Ad Hoc Consumer Electronics Goods Analysis

Retrieve data using text query(SELECT, WHERE, DISTINCT, LIKE)

```
SELECT market, fiscal_year, freight_pct
FROM fact_freight_cost;

SELECT *
FROM fact_gross_price
WHERE fiscal_year = 2020;

SELECT DISTINCT market
FROM fact_freight_cost;

SELECT product
FROM dim_product
WHERE product LIKE 'AQ MB%';
```

For data Filtering, aggregation, joins(Joins, Group by, Order by, Having, Between, In, If Clauses)

```
SELECT c.customer, d.market, p.pre_invoice_discount_pct
FROM fact_pre_invoice_deductions p
JOIN dim_customer c ON p.customer_code =
c.customer_code
JOIN fact_freight_cost d ON c.market = d.market AND
p.fiscal_year = d.fiscal_year;

SELECT market, COUNT(*) AS
num_customers
FROM dim_customer
GROUP BY market;

SELECT product_code, gross_price
FROM fact_gross_price
WHERE fiscal_year = 2022
ORDER BY gross_price DESC;

SELECT market, COUNT(*) AS count_customers
FROM dim_customer
GROUP BY market
HAVING COUNT(*) > 3;
```

```
SELECT *
FROM fact_freight_cost
WHERE freight_pct BETWEEN 0.02 AND
0.03;
```

```
SELECT *
FROM fact_gross_price
WHERE fiscal_year IN (2020, 2021);
```

```
SELECT *
FROM dim_customer
LIMIT 5;
```

```
SELECT *
FROM dim_customer
LIMIT 5 OFFSET 5;
```

```
SELECT customer_code,
    IF(pre_invoice_discount_pct > 0.2, 'High',
'Low') AS discount_level
FROM fact_pre_invoice_deductions
WHERE fiscal_year = 2022;
```

```
SELECT customer_code,
    CASE
      WHEN pre_invoice_discount_pct > 0.2
THEN 'High'
      WHEN pre_invoice_discount_pct > 0.1
THEN 'Medium'
      ELSE 'Low'
    END AS discount_category
FROM fact_pre_invoice_deductions
WHERE fiscal_year = 2022;
```

```
SELECT customer_code,
      CASE
        WHEN pre_invoice_discount_pct > 0.2
THEN 'High'
        WHEN pre_invoice_discount_pct > 0.1
THEN 'Medium'
        ELSE 'Low'
      END AS discount_category
FROM fact_pre_invoice_deductions
WHERE fiscal_year = 2022;
```

```
SELECT
    d.calender_date,
    YEAR(d.calender_date) AS calendar_year,
    dp.product,
    gp.gross_price
FROM dim_date d
JOIN fact_gross_price gp ON gp.fiscal_year = YEAR(d.calender_date)
JOIN dim_product dp ON gp.product_code = dp.product_code
WHERE dp.category = 'Graphic Card'
ORDER BY d.calender_date
LIMIT 10;
```

```
SELECT
    c.customer,
    p.fiscal_year,
    p.pre_invoice_discount_pct,
    CASE
        WHEN p.fiscal_year = CURYEAR() THEN 'Current Year'
        WHEN p.fiscal_year = CURYEAR() - 1 THEN 'Last Year'
        ELSE 'Older'
    END AS year_label
FROM fact_pre_invoice_deductions p
JOIN dim_customer c ON p.customer_code = c.customer_code
WHERE p.fiscal_year BETWEEN CURYEAR() - 2 AND CURYEAR()
ORDER BY c.customer
LIMIT 10;
```

Custom Function: `get_fiscal_year()`

```
-- Returns fiscal year for a given date. Assumes fiscal year starts
in September.

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
```

SQL Query

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

Stored Procedure – `get_monthly_gross_sales_for_customer`

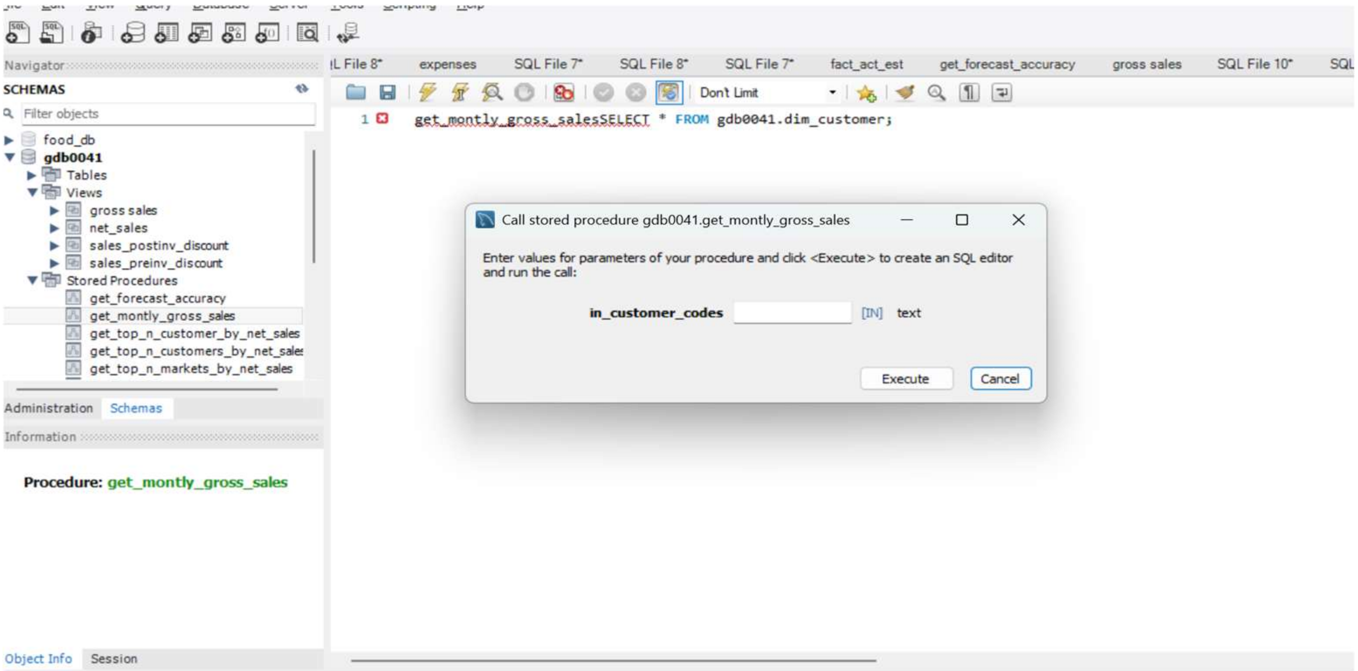
To improve scalability and reusability of monthly gross sales reporting, I created a stored procedure that accepts **multiple customer codes** and returns their **aggregated monthly gross sales**.

Purpose

This stored procedure enables dynamic gross sales reporting across one or more customers without rewriting the query logic. Useful for both **manual analysis** and **automated pipelines**.

SQL Code

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



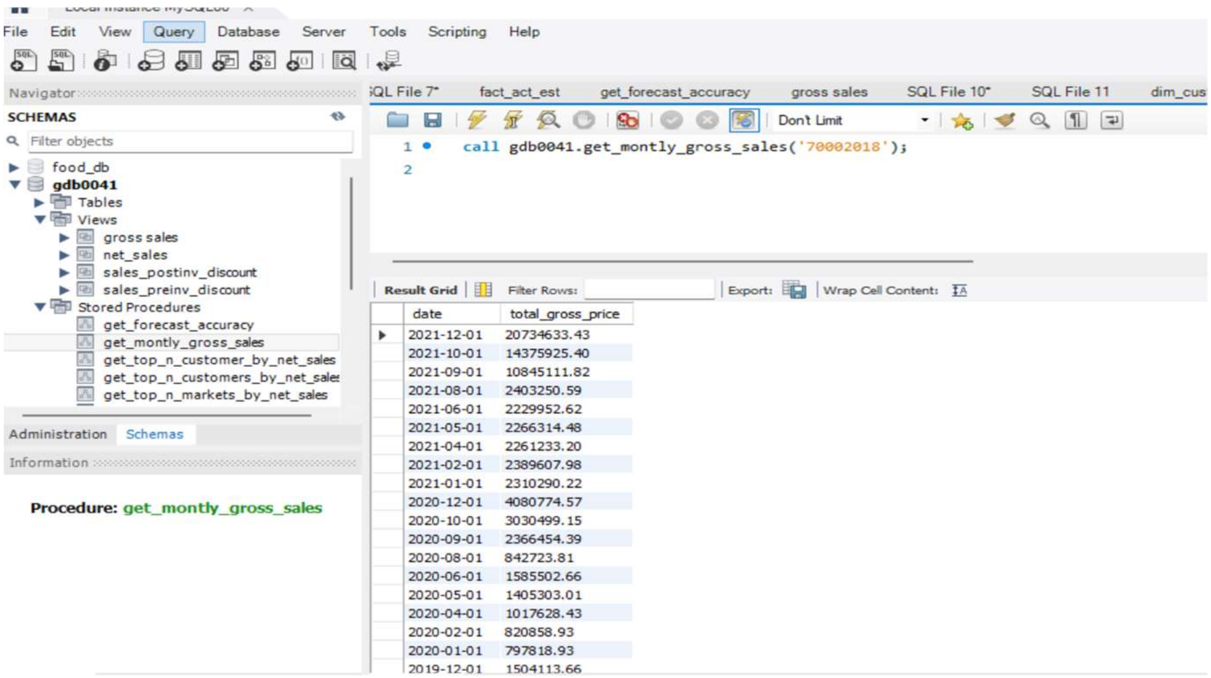
```
get_montly_gross_salesSELECT * FROM gdb0041.dim_customer;
```

Call stored procedure gdb0041.get_montly_gross_sales

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

in_customer_codes [IN] text

Procedure: get_montly_gross_sales



```
call gdb0041.get_montly_gross_sales('70002018');
```

date	total_gross_price
2021-12-01	20734633.43
2021-10-01	14375925.40
2021-09-01	10845111.82
2021-08-01	2403250.59
2021-06-01	2229952.62
2021-05-01	2266314.48
2021-04-01	2261233.20
2021-02-01	2389607.98
2021-01-01	2310290.22
2020-12-01	4080774.57
2020-10-01	3030499.15
2020-09-01	2366454.39
2020-08-01	842723.81
2020-06-01	1585502.66
2020-05-01	1405303.01
2020-04-01	1017628.43
2020-02-01	820858.93
2020-01-01	797818.93
2019-12-01	1504113.66

Procedure: get_montly_gross_sales

Create a stored procedure that can determine the market badge based on the following logic:

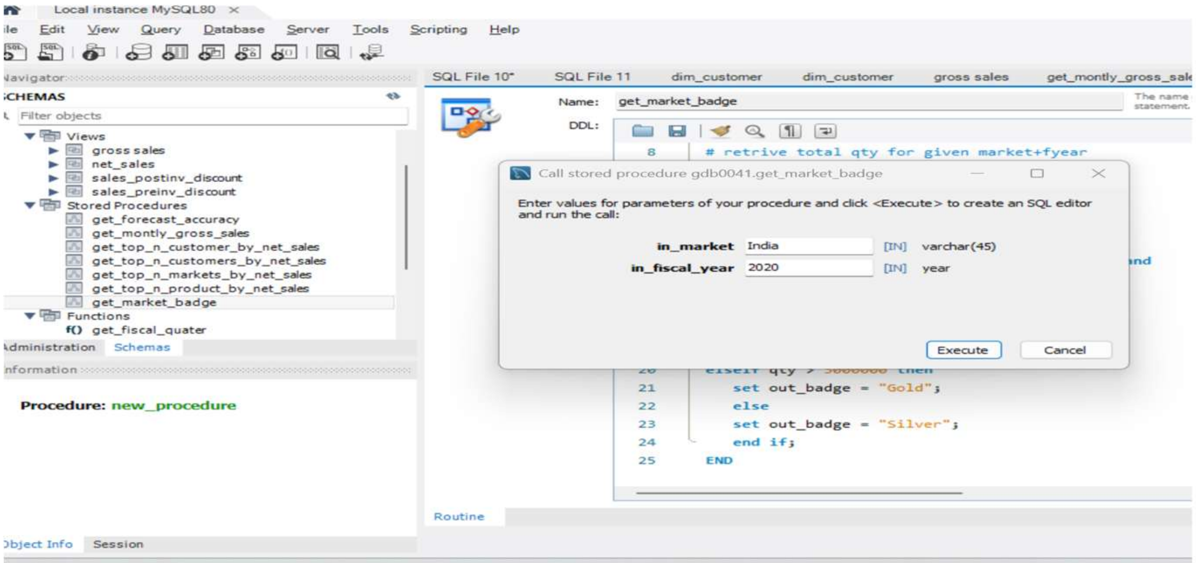
If total sold quantity > 5 million that market is considered Gold else it is Silver.

SQL Code

```
CREATE PROCEDURE get_market_badge(
    IN in_fiscal_year YEAR,
    IN in_market VARCHAR(45),
    OUT out_badge VARCHAR(45)
)
BEGIN
    DECLARE qty INT DEFAULT 0;

    -- Set default market to India if input is empty
    IF in_market = '' THEN
        SET in_market = 'India';
    END IF;
    -- Get total sold quantity for given market and fiscal year
    SELECT
        SUM(s.sold_quantity) INTO qty
    FROM
        fact_sales_monthly s
    JOIN
        dim_customer c ON s.customer_code = c.customer_code
    WHERE  get_fiscal_year(s.date) = in_fiscal_year
        AND c.market = in_market;

    -- Classify market badge based on quantity
    IF qty > 5000000 THEN
        SET out_badge = 'Gold';
    ELSE
        SET out_badge = 'Silver';
    END IF;
END
```



Name: get_market_badge

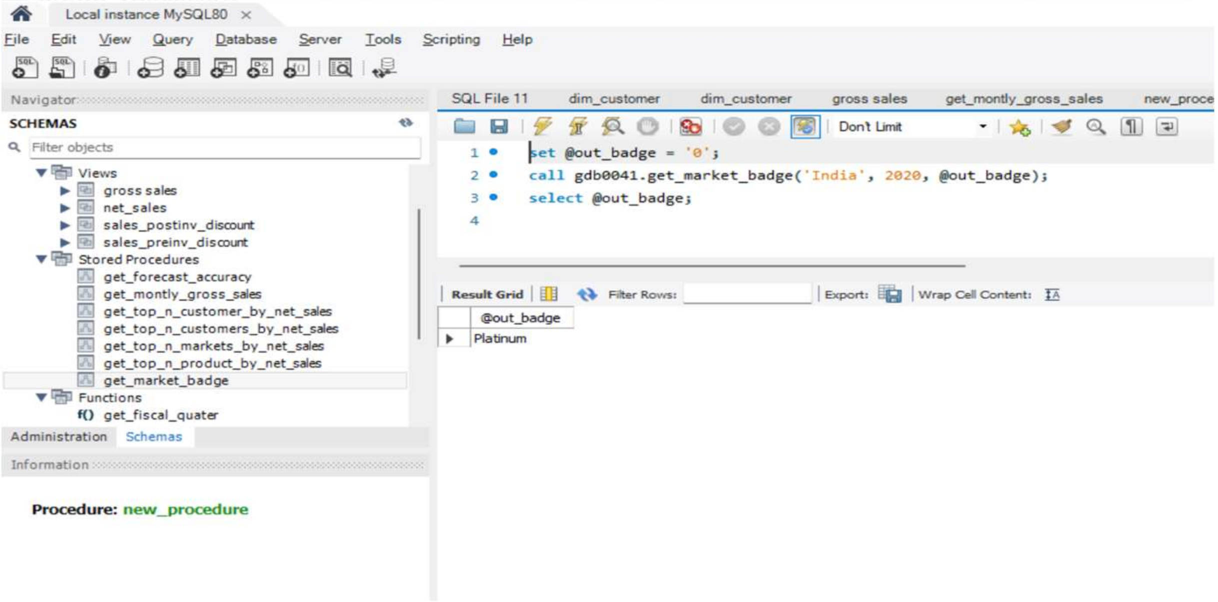
DDL:

```
# retrive total qty for given market+fyear
elseif qty > 5000000 then
    set out_badge = "Gold";
    else
    set out_badge = "Silver";
    end if;
END
```

Call stored procedure gdb0041.get_market_badge

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

in_market	India	[IN]	varchar(45)
in_fiscal_year	2020	[IN]	year

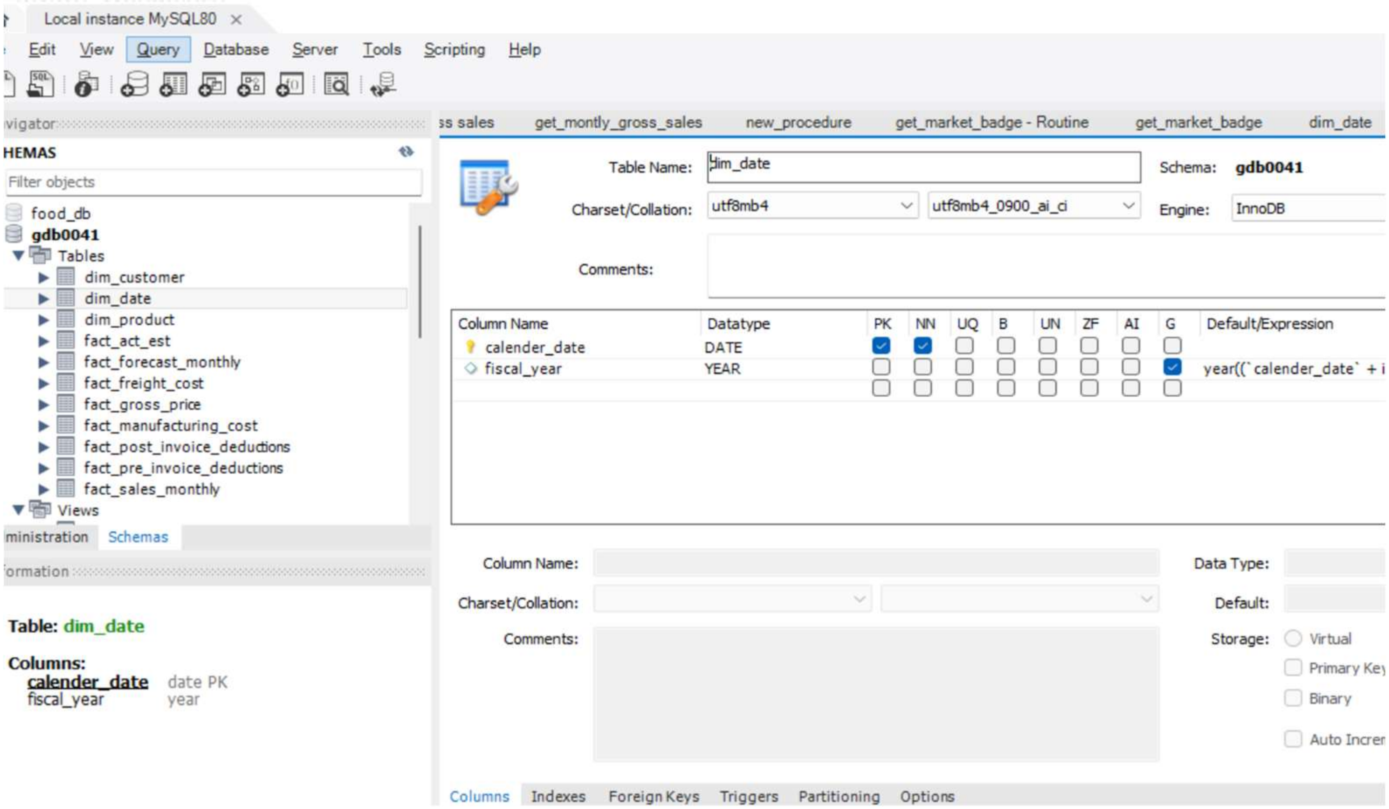


```
set @out_badge = '0';
call gdb0041.get_market_badge('India', 2020, @out_badge);
select @out_badge;
```

	@out_badge
▶	Platinum

Performance Improvement #1: Add `dim_date` Table

By introducing the `dim_date` table, we avoid applying functions directly on date columns in the `WHERE` clause and improve join logic across time-based tables. This enhances query performance and supports better filtering and aggregation.



- food_db
- gdb0041
  - Tables
    - dim_customer
    - dim_date
    - dim_product
    - fact_act_est
    - fact_forecast_monthly
    - fact_freight_cost
    - fact_gross_price
    - fact_manufacturing_cost
    - fact_post_invoice_deductions
    - fact_pre_invoice_deductions
    - fact_sales_monthly
  - Views

Table Name: dim_date Schema: **gdb0041**

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
calender_date	DATE	☑	☑	□	□	□	□	□	□	
fiscal_year	YEAR	□	□	□	□	□	□	□	☑	year((`calender_date` + i
		□	□	□	□	□	□	□	□	

Column Name:

Charset/Collation:

Comments:

Data Type:

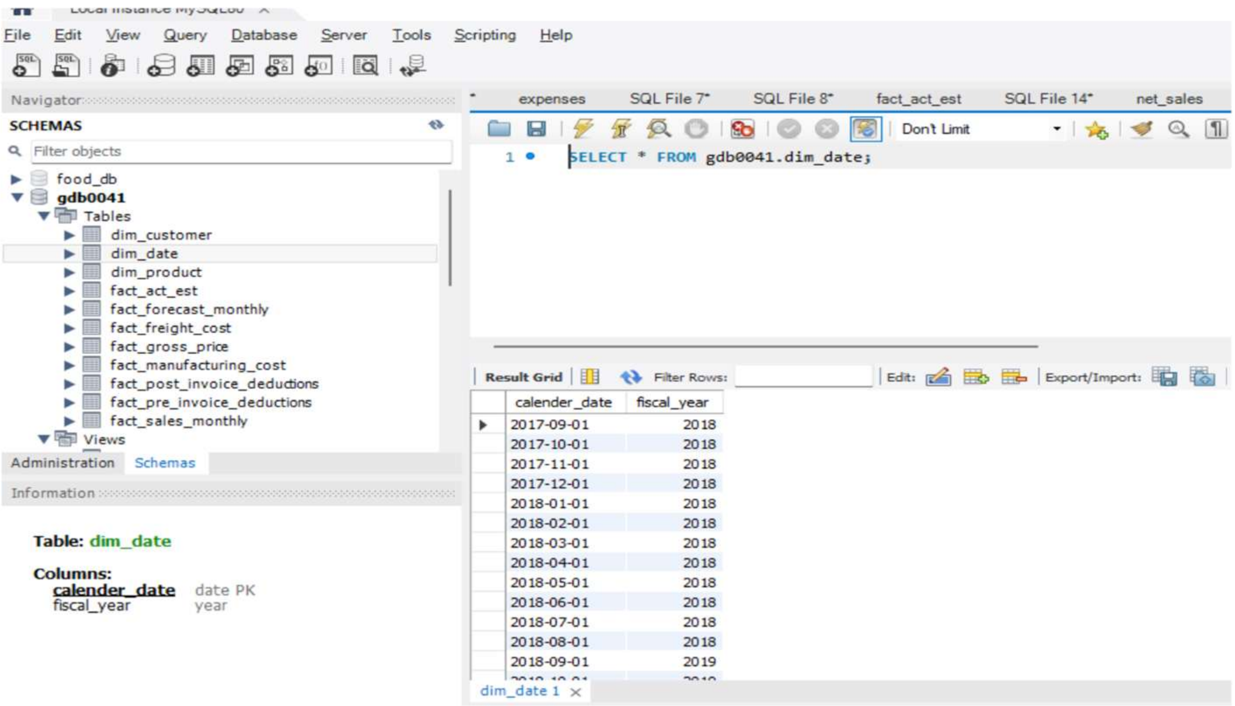
Default:

Storage: ○ Virtual □ Primary Key □ Binary □ Auto Increr

Table: dim_date

Columns:

calender_date	date PK
fiscal_year	year



- food_db
- gdb0041
  - Tables
    - dim_customer
    - dim_date
    - dim_product
    - fact_act_est
    - fact_forecast_monthly
    - fact_freight_cost
    - fact_gross_price
    - fact_manufacturing_cost
    - fact_post_invoice_deductions
    - fact_pre_invoice_deductions
    - fact_sales_monthly
  - Views

```
SELECT * FROM gdb0041.dim_date;
```

calender_date	fiscal_year
2017-09-01	2018
2017-10-01	2018
2017-11-01	2018
2017-12-01	2018
2018-01-01	2018
2018-02-01	2018
2018-03-01	2018
2018-04-01	2018
2018-05-01	2018
2018-06-01	2018
2018-07-01	2018
2018-08-01	2018
2018-09-01	2019

Table: dim_date

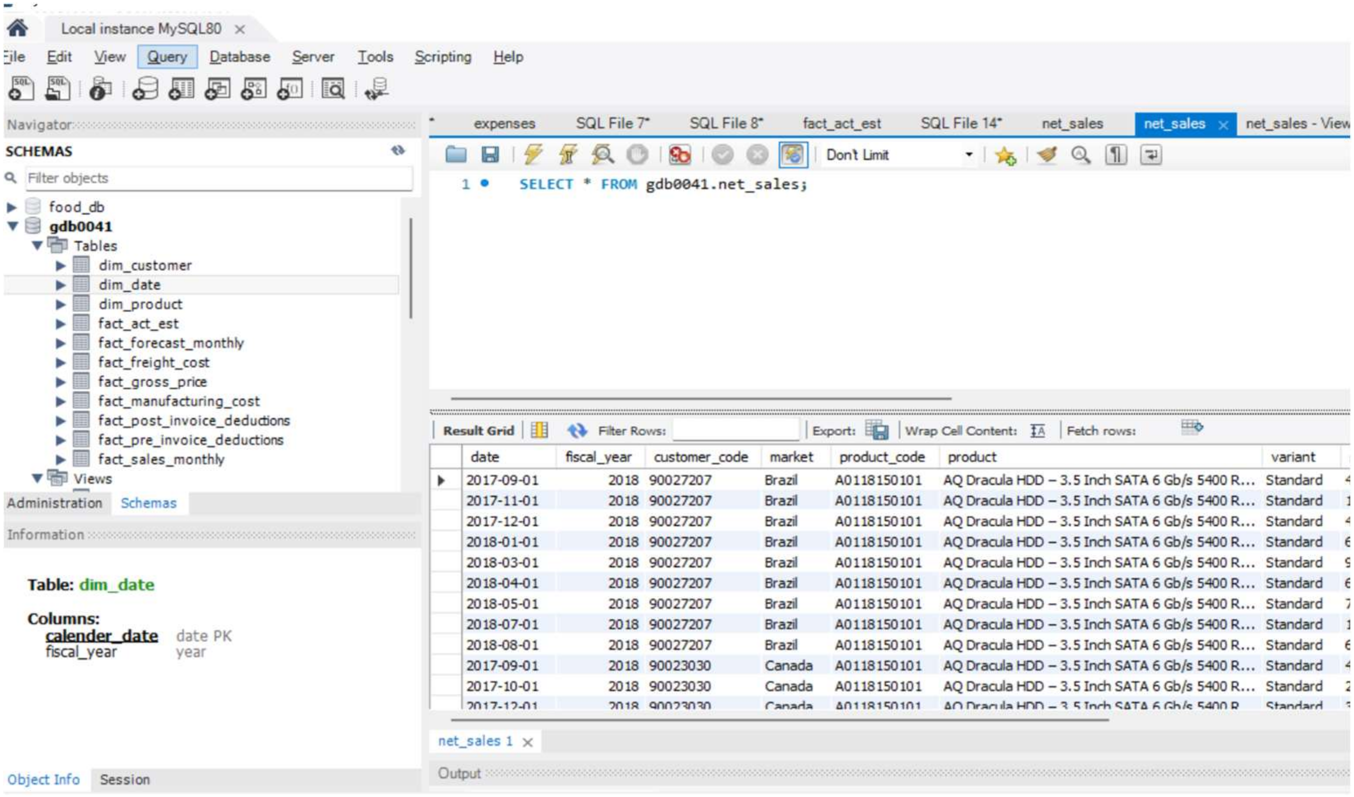
Columns:

calender_date	date PK
fiscal_year	year

Performance Improvement #2: Add `fiscal_year` Column in `fact_sales_monthly`

Adding a `fiscal_year` column to the `fact_sales_monthly` table simplifies joins with other fact tables and eliminates the need for a date dimension join in certain use cases, improving performance and clarity.

```
SELECT
    s.date,
    s.product_code,
    p.product,
    p.variant,
    s.sold_quantity,
    ROUND(g.gross_price, 2) AS gross_price,
    ROUND(g.gross_price * s.sold_quantity, 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.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 pre.fiscal_year = s.fiscal_year
WHERE
    s.fiscal_year = 2021
LIMIT 1000000;
```



```
SELECT * FROM gdb0041.net_sales;
```

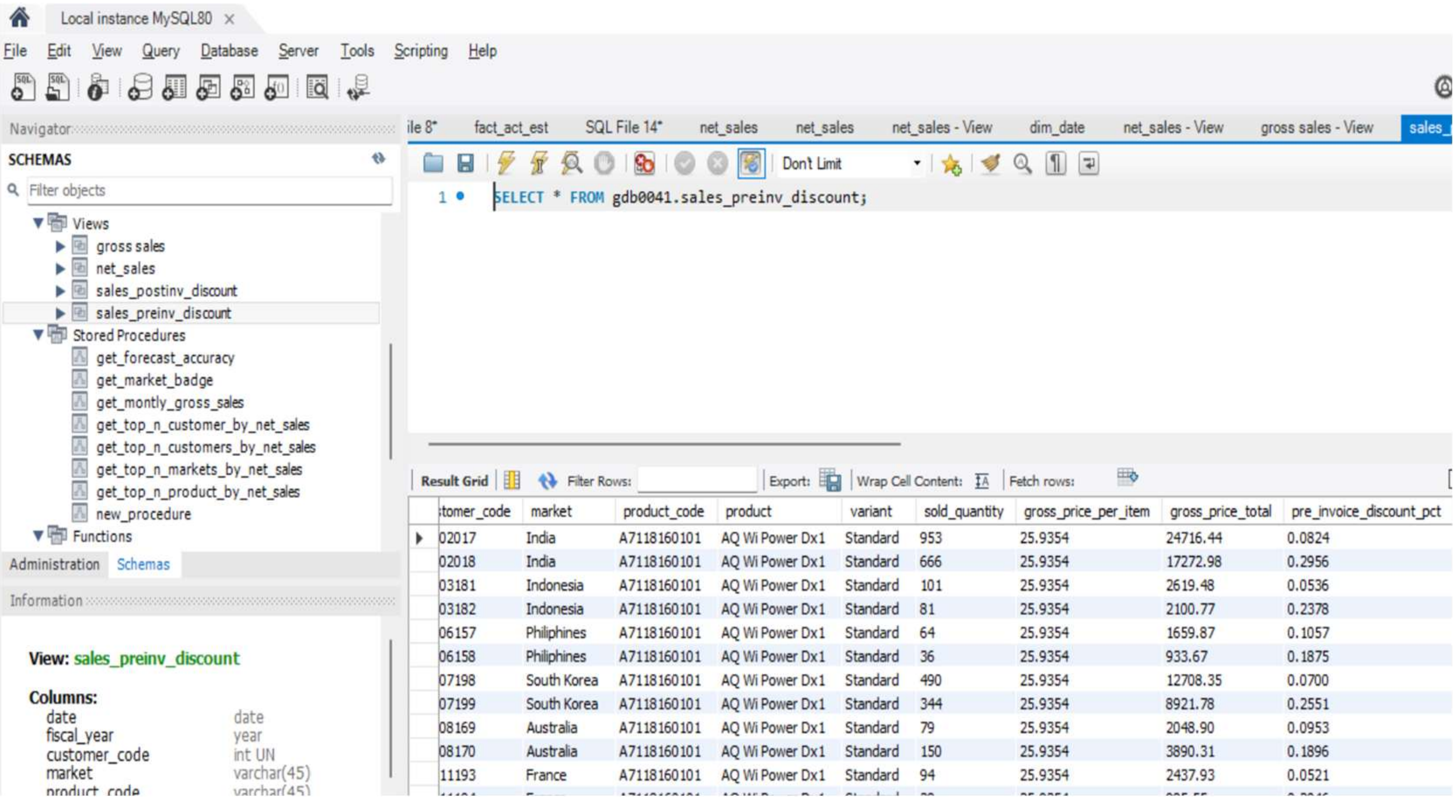
Table: dim_date

Columns:
calender_date date PK
fiscal_year year

date	fiscal_year	customer_code	market	product_code	product	variant
2017-09-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2017-11-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2017-12-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2018-01-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2018-03-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2018-04-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2018-05-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2018-07-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2018-08-01	2018	90027207	Brazil	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2017-09-01	2018	90023030	Canada	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2017-10-01	2018	90023030	Canada	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard
2017-12-01	2018	90023030	Canada	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 R	Standard

Create `sales_preinv_discount` View

```
CREATE VIEW sales_preinv_discount AS
SELECT
    s.date,
    s.fiscal_year,
    s.customer_code,
    c.market,
    s.product_code,
    p.product,
    p.variant,
    s.sold_quantity,
    ROUND(g.gross_price, 2) AS gross_price,
    ROUND(g.gross_price * s.sold_quantity, 2) AS gross_price_total,
    pre.pre_invoice_discount_pct
FROM
    fact_sales_monthly s
JOIN dim_customer c
    ON c.customer_code = s.customer_code
JOIN dim_product p
    ON s.product_code = p.product_code
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 pre.fiscal_year =
s.fiscal_year;
```



```
SELECT * FROM gdb0041.sales_preinv_discount;
```

tomer_code	market	product_code	product	variant	sold_quantity	gross_price_per_item	gross_price_total	pre_invoice_discount_pct
02017	India	A7118160101	AQ Wi Power Dx1	Standard	953	25.9354	24716.44	0.0824
02018	India	A7118160101	AQ Wi Power Dx1	Standard	666	25.9354	17272.98	0.2956
03181	Indonesia	A7118160101	AQ Wi Power Dx1	Standard	101	25.9354	2619.48	0.0536
03182	Indonesia	A7118160101	AQ Wi Power Dx1	Standard	81	25.9354	2100.77	0.2378
06157	Philiphines	A7118160101	AQ Wi Power Dx1	Standard	64	25.9354	1659.87	0.1057
06158	Philiphines	A7118160101	AQ Wi Power Dx1	Standard	36	25.9354	933.67	0.1875
07198	South Korea	A7118160101	AQ Wi Power Dx1	Standard	490	25.9354	12708.35	0.0700
07199	South Korea	A7118160101	AQ Wi Power Dx1	Standard	344	25.9354	8921.78	0.2551
08169	Australia	A7118160101	AQ Wi Power Dx1	Standard	79	25.9354	2048.90	0.0953
08170	Australia	A7118160101	AQ Wi Power Dx1	Standard	150	25.9354	3890.31	0.1896
11193	France	A7118160101	AQ Wi Power Dx1	Standard	94	25.9354	2437.93	0.0521

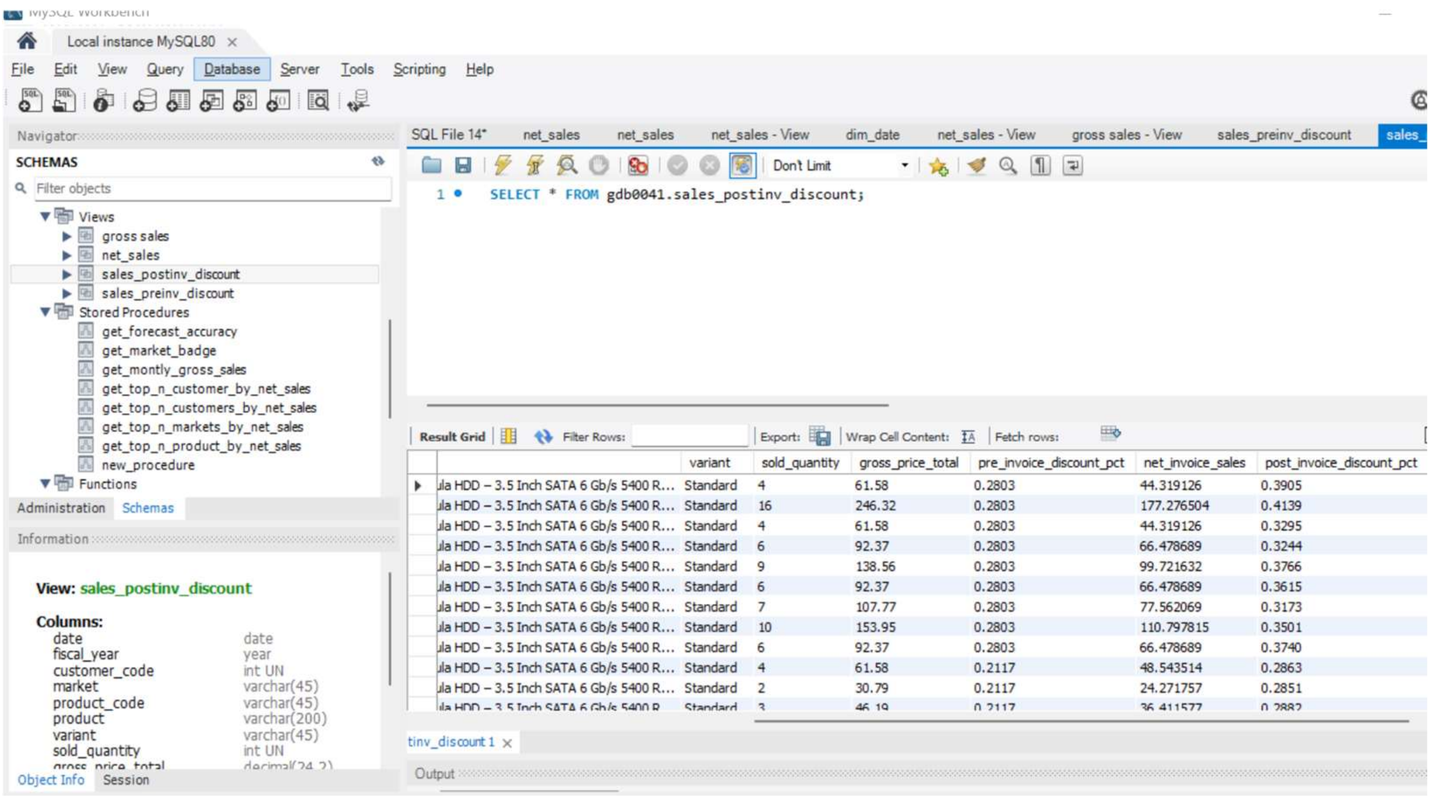
View: sales_preinv_discount

Columns:

date	date
fiscal_year	year
customer_code	int UN
market	varchar(45)
product_code	varchar(45)

Create `sales_postinv_discount` View

```sql
CREATE VIEW sales_postinv_discount AS
SELECT
    s.date,
    s.fiscal_year,
    s.customer_code,
    s.market,
    s.product_code,
    s.product,
    s.variant,
    s.sold_quantity,
    s.gross_price,
    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;
```



```sql
SELECT * FROM gdb0041.sales_postinv_discount;
```

	variant	sold_quantity	gross_price_total	pre_invoice_discount_pct	net_invoice_sales	post_invoice_discount_pct
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	4	61.58	0.2803	44.319126	0.3905
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	16	246.32	0.2803	177.276504	0.4139
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	4	61.58	0.2803	44.319126	0.3295
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	6	92.37	0.2803	66.478689	0.3244
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	9	138.56	0.2803	99.721632	0.3766
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	6	92.37	0.2803	66.478689	0.3615
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	7	107.77	0.2803	77.562069	0.3173
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	10	153.95	0.2803	110.797815	0.3501
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	6	92.37	0.2803	66.478689	0.3740
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	4	61.58	0.2117	48.543514	0.2863
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R...	Standard	2	30.79	0.2117	24.271757	0.2851
ıla HDD – 3.5 Inch SATA 6 Gb/s 5400 R	Standard	3	46.19	0.2117	36.411577	0.2882

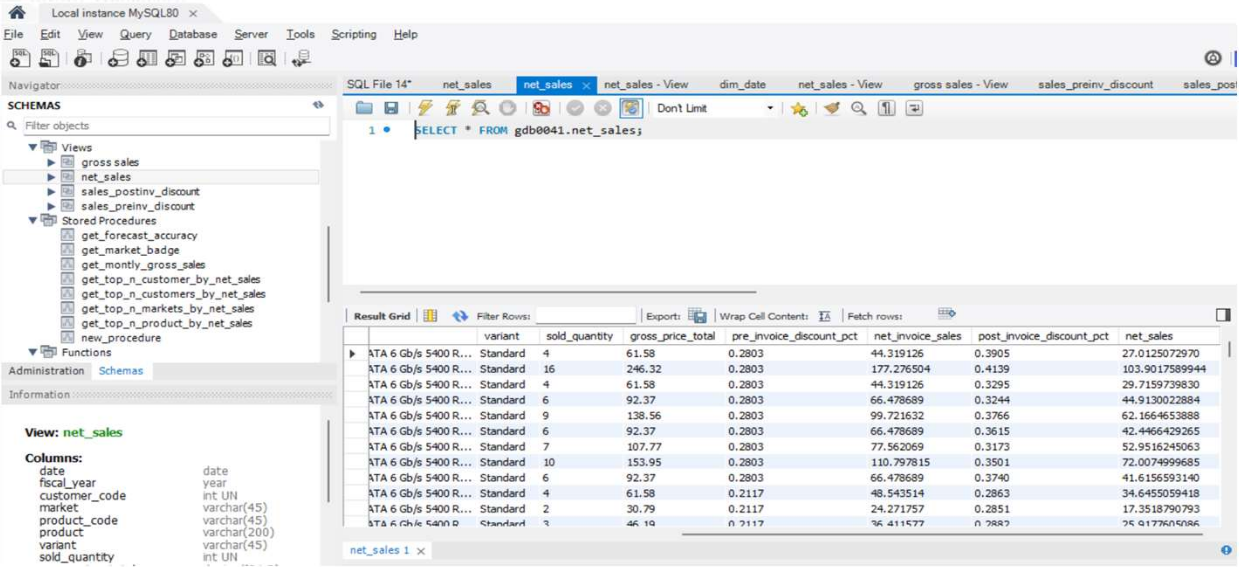
View: sales_postinv_discount

Columns:

date	date
fiscal_year	year
customer_code	int UN
market	varchar(45)
product_code	varchar(45)
product	varchar(200)
variant	varchar(45)
sold_quantity	int UN
gross_price_total	decimal(24,2)

Calculate Final Net Sales

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



```
SELECT * FROM gdb0041.net_sales;
```

View: net_sales

Columns:

date	date
fiscal_year	year
customer_code	int UN
market	varchar(45)
product_code	varchar(45)
product	varchar(200)
variant	varchar(45)
sold_quantity	int UN

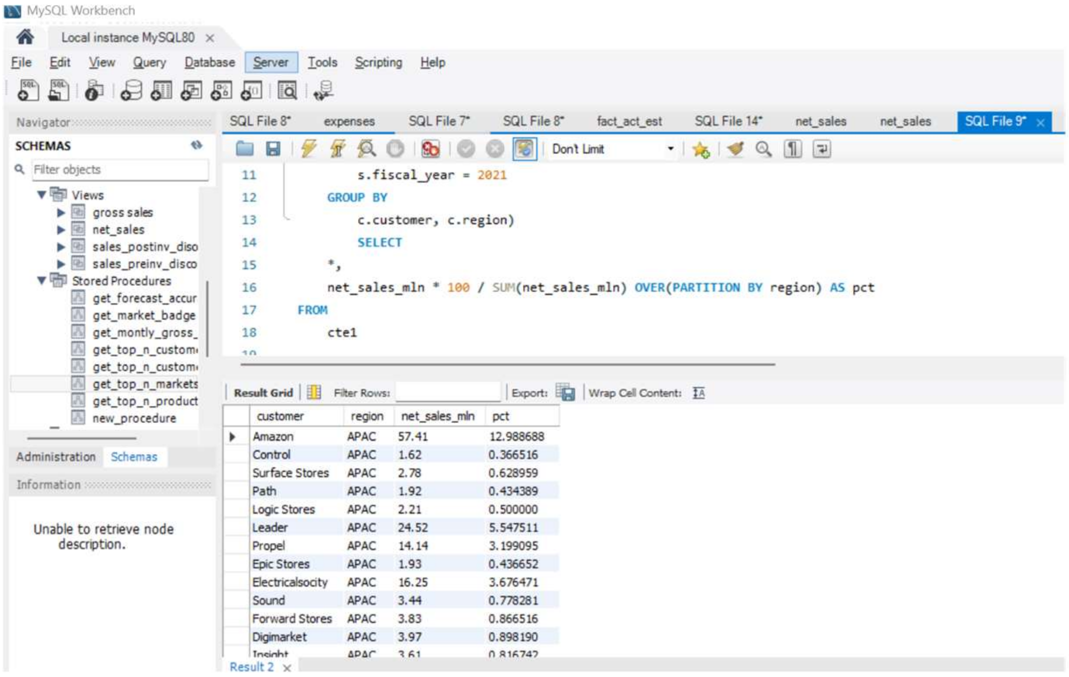
	variant	sold_quantity	gross_price_total	pre_invoice_discount_pct	net_invoice_sales	post_invoice_discount_pct	net_sales
ATA 6 Gb/s 5400 R...	Standard	4	61.58	0.2803	44.319126	0.3905	27.0125072970
ATA 6 Gb/s 5400 R...	Standard	16	246.32	0.2803	177.276504	0.4139	103.9017589944
ATA 6 Gb/s 5400 R...	Standard	4	61.58	0.2803	44.319126	0.3295	29.7159739830
ATA 6 Gb/s 5400 R...	Standard	6	92.37	0.2803	66.478689	0.3244	44.9130022884
ATA 6 Gb/s 5400 R...	Standard	9	138.56	0.2803	99.721632	0.3766	62.1664653888
ATA 6 Gb/s 5400 R...	Standard	6	92.37	0.2803	66.478689	0.3615	42.4466429265
ATA 6 Gb/s 5400 R...	Standard	7	107.77	0.2803	77.562069	0.3173	52.9516245063
ATA 6 Gb/s 5400 R...	Standard	10	153.95	0.2803	110.797815	0.3501	72.0074999685
ATA 6 Gb/s 5400 R...	Standard	6	92.37	0.2803	66.478689	0.3740	41.6156593140
ATA 6 Gb/s 5400 R...	Standard	4	61.58	0.2117	48.543514	0.2863	34.6455059418
ATA 6 Gb/s 5400 R...	Standard	2	30.79	0.2117	24.271757	0.2851	17.3518790793
ATA 6 Gb/s 5400 R	Standard	3	46.19	0.2117	36.411577	0.2882	25.9177605086

Generate a Region-wise % Net Sales Breakdown by Customers.

This query generates a **region-wise percentage breakdown of net sales by customers for a given fiscal year.
It supports **regional financial analysis** by showing how much each customer contributes to the total sales in their respective region.**

SQL Query

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



```
            s.fiscal_year = 2021
        GROUP BY
            c.customer, c.region)
            SELECT
        *,
        net_sales_mln * 100 / SUM(net_sales_mln) OVER(PARTITION BY region) AS pct
    FROM
        cte1
```

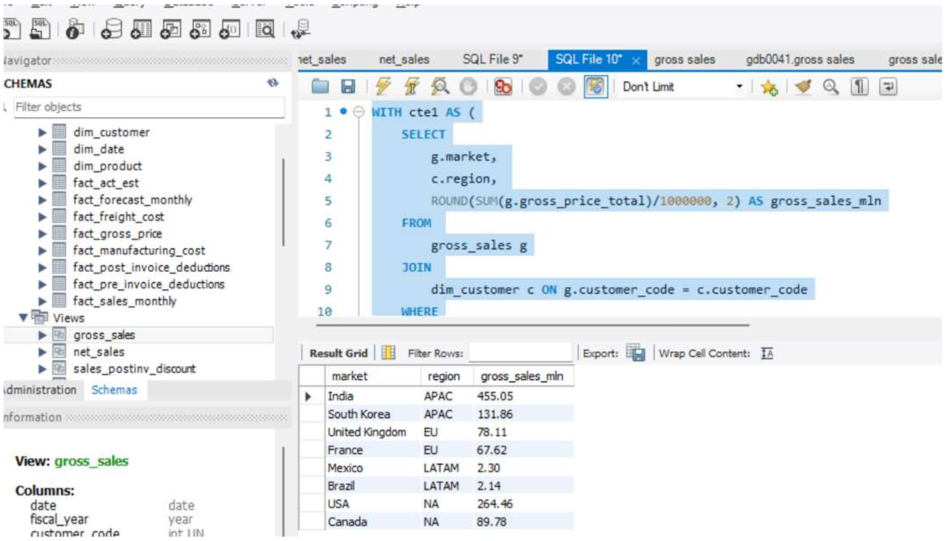
customer	region	net_sales_mln	pct
Amazon	APAC	57.41	12.988688
Control	APAC	1.62	0.366516
Surface Stores	APAC	2.78	0.628959
Path	APAC	1.92	0.434389
Logic Stores	APAC	2.21	0.500000
Leader	APAC	24.52	5.547511
Propel	APAC	14.14	3.199095
Epic Stores	APAC	1.93	0.436652
Electricalsocity	APAC	16.25	3.676471
Sound	APAC	3.44	0.778281
Forward Stores	APAC	3.83	0.866516
Digimarket	APAC	3.97	0.898190
Insight	APAC	3.61	0.816742

Retrieve the Top 2 Markets in Every Region by Gross Sales.

This query identifies the **top 2 performing markets within each region based on their **gross sales** for the fiscal year **2021**.
It uses the `DENSE_RANK()` window function to handle ties in rankings accurately.**

SQL Query

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



```
WITH cte1 AS (
    SELECT
        g.market,
        c.region,
        ROUND(SUM(g.gross_price_total)/1000000, 2) AS gross_sales_mln
    FROM
        gross_sales g
    JOIN
        dim_customer c ON g.customer_code = c.customer_code
    WHERE
```

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

Generate an Aggregate Forecast Accuracy Report for Customers for a Given Fiscal Year.

Forecast Accuracy Calculation

Using the helper table, the query computes:

- **Total sold quantity** and **forecast quantity** per customer
- **Net forecast error** and **net error percentage**
- **Absolute forecast error** and **absolute error percentage**
- **Forecast accuracy** as:
  `Forecast Accuracy = 100 - Absolute Error %` (capped at 100%)

Create helper table combining actual sales and forecast quantities

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

Calculate forecast accuracy metrics per customer for fiscal year 2021
WITH forecast_err_table

```
(
   SELECT
      customer_code,
      SUM(sold_quantity) AS total_sold_quantity,
      SUM(forecast_quantity) AS total_forecast_quantity,
      SUM(forecast_quantity - sold_quantity) AS net_err,
      SUM(forecast_quantity - sold_quantity) * 100 / SUM(forecast_quantity)
AS net_err_pct,
      SUM(ABS(forecast_quantity - sold_quantity)) AS abs_err,
      SUM(ABS(forecast_quantity - sold_quantity)) * 100 /
SUM(forecast_quantity) AS abs_err_pct
   FROM
      fact_act_est
   WHERE
      fiscal_year = 2021
   GROUP BY
      customer_code
)
SELECT
   e.*,
   c.market,
   c.customer,
   IF(abs_err_pct > 100, 0, 100 - abs_err_pct) AS forecast_accuracy
FROM
   forecast_err_table e
JOIN
   dim_customer c
USING
   (customer_code)
ORDER BY
   forecast_accuracy DESC;
```