AtliQ Hardwares Finance and Supply Chain Analytics Project

SQL queries

Views

1. Net sales view

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `net_sales` AS
    SELECT
        `sales_postinv_discount`.`date` AS `date`,
        `sales_postinv_discount`.`fiscal_year` AS `fiscal_year`,
        `sales_postinv_discount`.`customer_code` AS `customer_code`,
        `sales_postinv_discount`.`market` AS `market`,
        `sales_postinv_discount`.`product_code` AS `product_code`,
        `sales_postinv_discount`.`product` AS `product`,
        `sales_postinv_discount`.`variant` AS `variant`,
        `sales_postinv_discount`.`sold_quantity` AS `sold_quantity`,
        `sales_postinv_discount`.`gross_price_total` AS `gross_price_total`,
        `sales_postinv_discount`.`pre_invoice_discount_pct` AS `pre_invoice_discount_pct`,
        `sales_postinv_discount`.`net_invoice_sales` AS `net_invoice_sales`,
        `sales_postinv_discount`.`post_invoice_discount_pct` AS `post_invoice_discount_pct`,
        ((1 - `sales_postinv_discount`.`post_invoice_discount_pct`) *
`sales_postinv_discount`.`net_invoice_sales`) AS `net_sales`
    FROM
        `sales_postinv_discount`
```

2. Sales_pre_invoice_deductions

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
```

```
  SQL SECURITY DEFINER
VIEW `sales_preinv_discount` AS
  SELECT
    `s`.`date` AS `date`,
    `s`.`fiscal_year` AS `fiscal_year`,
    `s`.`customer_code` AS `customer_code`,
    `c`.`market` AS `market`,
    `s`.`product_code` AS `product_code`,
    `p`.`product` AS `product`,
    `p`.`variant` AS `variant`,
    `s`.`sold_quantity` AS `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` AS `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` `pre` ON (((`pre`.`customer_code` =
`s`.`customer_code`)
      AND (`pre`.`fiscal_year` = `s`.`fiscal_year`))))
```

3. Sales_post_invoice_dedctions

```
    CREATE
      ALGORITHM = UNDEFINED
      DEFINER = `root`@`localhost`
      SQL SECURITY DEFINER
    VIEW `sales_postinv_discount` AS
      SELECT
        `s`.`date` AS `date`,
```

```
    `s`.`fiscal_year` AS `fiscal_year`,
    `s`.`customer_code` AS `customer_code`,
    `s`.`market` AS `market`,
    `s`.`product_code` AS `product_code`,
    `s`.`product` AS `product`,
    `s`.`variant` AS `variant`,
    `s`.`sold_quantity` AS `sold_quantity`,
    `s`.`gross_price_total` AS `gross_price_total`,
    `s`.`pre_invoice_discount_pct` AS `pre_invoice_discount_pct`,
    (`s`.`gross_price_total` - (`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`))))
```

Stored_Procedures

1. **Top_n_products by net sales**

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `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
```

2. **Top n products per division by qty**

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`get_top_n_products_per_division_by_qty_sold`(
    in_fiscal_year int,
    in_top_n int
)
```

```
BEGIN
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=in_fiscal_year
 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<=in_top_n;
END
```

3. Top n markets by net sales

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_top_n_markets_by_net_sales`(
 in_fiscal_year int,
 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
```

4. Top n customers by net sales

```
CREATE DEFINER=`root`@`localhost` 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
          join dim_customer c
          on s.customer_code=c.customer_code
          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
```

5. Monthly gross sales by customers

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_monthly_gross_sales_for_customers`(
    in_customer_codes text
)
BEGIN
select
     s.date,
     sum(round(g.gross_price*s.sold_quantity,2)) as gross_price_total
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;
END
```

6. Forecast accuracy

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_forecast_accuracy`(
          in_fiscal_year INT
```

```
        )
BEGIN
with forecast_err_table as
(
select
                s.customer_code as customer_code,
                c.customer as customer_name,
                c.market as market,
                sum(s.sold_quantity) as total_sold_qty,
                sum(s.forecast_quantity) as total_forecast_qty,
                sum(s.forecast_quantity-s.sold_quantity) as net_error,
                round(sum(s.forecast_quantity-
s.sold_quantity)*100/sum(s.forecast_quantity),1) as net_error_pct,
                sum(abs(s.forecast_quantity-s.sold_quantity)) as abs_error,
                round(sum(abs(s.forecast_quantity-
sold_quantity))*100/sum(s.forecast_quantity),2) as abs_error_pct
from fact_act_est s
join dim_customer c
        on s.customer_code = c.customer_code
where s.fiscal_year=in_fiscal_year
group by customer_code
)
select
        *,
if (abs_error_pct > 100, 0, 100.0 - abs_error_pct) as forecast_accuracy
from forecast_err_table
order by forecast_accuracy desc;
END
```

User defined functions

1. Get fiscal year

```
CREATE DEFINER=`root`@`localhost` FUNCTION `get_fiscal_year`(
    calender_date date
) RETURNS int
   DETERMINISTIC
BEGIN
   declare fiscal_year int;
   set fiscal_year = year(date_add(calender_date, interval 4 month));
return fiscal_year;
END
```

2. Get fiscal quarter

```
CREATE DEFINER=`root`@`localhost` FUNCTION `get_fiscal_quarter`(
   calender_date date
) RETURNS char(2) CHARSET utf8mb4
   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";
                else
           set qtr = "Q4";
    end case;
RETURN qtr;
END
```