Final document

# #Sl.	12
Last Update	@February 9, 2026 3:38 PM
Phase	Documentation
Status	In progress

▼ **Context:**

AtliQ Hardware experienced sustained financial losses in the Latin America region despite steady sales activity across global markets. The leadership team lacks timely, consolidated financial insights due to reliance on Excel-based reporting, limiting their ability to identify profit and cost drivers at scale

The objective of this project is to build a **reliable, data-driven Profit & Loss (P&L) view** using historical transactional data and supporting cost tables, in order to:

- Quantify **Gross Sales, Net Sales, Costs, and Profitability** over time
- Identify **which components of the P&L are contributing to losses**, with a specific focus on **Latin America**
- Compare **market-level performance** to distinguish between high-volume, high-value, and volatile markets
- Provide **actionable insights** to help management prioritize corrective actions that can improve profitability

This analysis covers historical data between **September 2017 and December 2021**, using validated fact and dimension tables from the enterprise data warehouse.

Success will be achieved if the analysis identifies **which P&L components (sold_quantity, price, discounts, freight, manufacturing cost, operating expenses etc.)** contribute most to the negative variance in Latin America compared to other regions or prior periods

▼ **Dataset**

This analysis uses two databases.

- gdb041 has dim_customer, dim_market, dim_product, fact_forecast_monthly, fact_sales_monthly. And,
- gdb056 has freight_cost, gross_price, manufacturing_cost, post_invoice_decution, pre_inovoice_deduction.

—*Fact_sales_monthly* will be regarded as FSM—

▼ **FSM_Sample Data Check:**

What Was Checked?

- Row Counts
- Available Columns

- Grain

How it was Checked?

- SQL Exploration: **Ref: 1. SQL_Explore_fact_sales_analysis**

▼ Code

```
/*
How many rows fact_sales_monthly have?
*/
SELECT count(*) FROM gdb041.fact_sales_monthly

/*
Check sample data
*/
SELECT * FROM gdb041.fact_sales_monthly LIMIT 1000

/*
The following query will help to understand to find the grain.
What each row in this dataset actually represents.
I am assuming that date+product_code+customer_code is the grain.
If the query return no rows. The assumption is validated.
*/
SELECT
    count(*) as row_count,
    `date`,
    product_code,
    customer_code
FROM
    gdb041.fact_sales_monthly
GROUP BY
    `date`,
    product_code,
    customer_code
HAVING
    ROW_COUNT>1

-- No rows return --
-- The grain date+product_code+customer_code
```

Findings:

- **Rows:** 1425706 rows
- **Columns:** 11 columns
- **Column List:** `'date', 'division', 'category', 'product_code', 'product', 'market', 'platform', 'channel', 'customer_code', 'customer_name', 'sold_quantity'`
- **Grain:** No duplicate records were found at the defined grain `date+product_code+customer_code`.

Decisions:

- Each row in *fact_sales_monthly* represents: A specific customer buys a specific product in a specific month

Notes:

- `sold_quantity` is the number of products being sold by a customer in a specific market in a specific month.
- This table captures **sales volume only.** Pricing, discounts, and cost components are handled in subsequent tables.

▼ **FSM_Data_type**

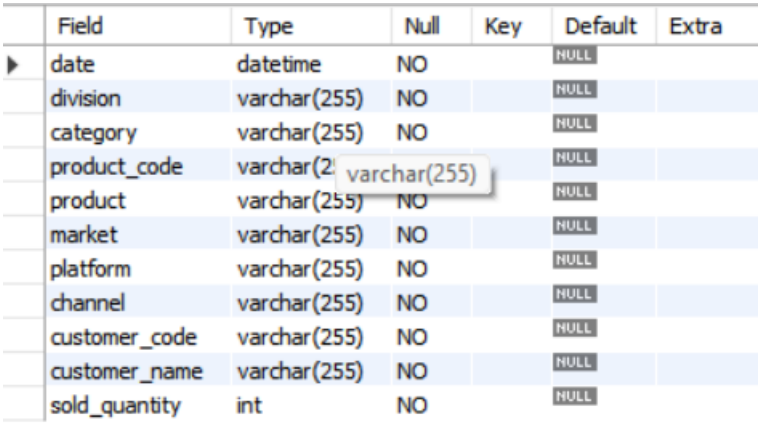
How it was checked?

SQL: Ref: 2. FSM_data_type

▼ Code

```
-- Check data type for all columns --
DESCRIBE fact_sales_monthly
```

Findings:



Field	Type	Null	Key	Default	Extra
date	datetime	NO		NULL	
division	varchar(255)	NO		NULL	
category	varchar(255)	NO		NULL	
product_code	varchar(2			NULL	
product	varchar(255)	NO		NULL	
market	varchar(255)	NO		NULL	
platform	varchar(255)	NO		NULL	
channel	varchar(255)	NO		NULL	
customer_code	varchar(255)	NO		NULL	
customer_name	varchar(255)	NO		NULL	
sold_quantity	int	NO		NULL	

Decision:

- No data type issues detected

▼ **FSM_Distinct Values Per column:**

How it was checked?

SQL: **Ref: 3. FSM_distinct_values**

▼ Code:

```
-- Check distinct values for all categorical column --

SELECT 'product' as column_name, COUNT(DISTINCT product) AS distinct
_count FROM gdb041.fact_sales_monthly
```

```
UNION ALL
SELECT 'customer_name', COUNT(DISTINCT customer_name) FROM gdb041.fa
ct_sales_monthly
UNION ALL
SELECT 'market', COUNT(DISTINCT market) FROM gdb041.fact_sales_month
ly
UNION ALL
SELECT 'channel', COUNT(DISTINCT channel) FROM gdb041.fact_sales_mon
thly
UNION ALL
SELECT 'platform',COUNT(DISTINCT platform)FROM gdb041.fact_sales_mon
thly
UNION ALL
SELECT 'division',COUNT(DISTINCT division)FROM gdb041.fact_sales_mon
thly
UNION ALL
SELECT 'category',COUNT(DISTINCT category)FROM gdb041.fact_sales_mon
thly;
```

Findings:

column_name	distinct_count
product	71
customer_name	75
market	27
channel	3
platform	2
division	3
category	14

Decision:

- `market, channel, platform, division, category` - They are low cardinal column. Can be used as a slicer in power BI.
- In our database, we already had *dim_product, dim_customer, dim_market*.
- The grain for FSM is `date+customer_code+product_code`
- The other descriptive column could be connected by using common keys.
- It would be easier to process data if redundant descriptive columns can be removed.
- At first check sample data for *dim_customer*

—*dim_customer* will be regarded as DC—

▼ **DC_Sample Data Check:**

What Was Checked?

- Row Counts
- Available Columns

- Grain
- Data type for all columns

How it was Checked?

SQL: Ref: 4. DC_Sample_data_check

▼ Code

```
/*
How many rows dim_customer have?
*/
SELECT count(*) FROM gdb041.dim_customer

/*
Check sample data
*/
SELECT * FROM gdb041.dim_customer LIMIT 10

/*
The following query will help to understand to find the grain.
What each row in this dataset actually represents.
I am assuming that customer_code is the grain.
If the query return no rows. The assumption is validated.
*/
SELECT
    count(*) as row_counts,
    customer_code
FROM
    gdb041.dim_customer
GROUP BY
    customer_code
HAVING
    ROW_COUNTS>1

-- No rows return --
-- The grain is customer_code --
 -- Check data type for all columns --
DESCRIBE dim_customer
```

Findings:

- **Rows:** 209 rows
- **Columns: 5** columns
- **Column List:** `customer, market, platform, channel, customer_code`
- **Grain:** No duplicate records were found at the defined grain `customer_code`.
- **Data type:** All column is varchar(255)

Decisions:

- Each row in dim_customer represents details for a specific customer_code

Notes:

- If referential integrity between FSM and DC is intact, descriptive columns (`customer, platform, channel`) can be removed.

▼ **Check referential integrity for customer_code between FSM and DC**

Checklist:

- **Grain:** FSM: `date+product_code+customer_code`, DC: `customer_code`
- **Data Type:** VARCHAR for both
- `customer_code` is unique in *dm_customer*

▼ **Whitespace_case_inconsistencies check for customer_code in FSM and DC**

How it was checked?

SQL: Ref: 5. Customer_code_Whitespace_case_inconsistencies

```
/* The purpose of this query is to find whitespaces and case inconsi
stencies in the customer_code columns*/
SELECT
    'fact_table' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT customer_code) AS examples
FROM gdb041.fact_sales_monthly
WHERE customer_code != UPPER(TRIM(customer_code))

UNION ALL

SELECT
    'customer_table' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT customer_code) AS examples
FROM gdb041.dim_customer
WHERE customer_code != UPPER(TRIM(customer_code));
```

Decision:

- No whitespace and case_inconsistencies detected

▼ **Null values for customer_Code in FSM and DC**

How it was checked?

SQL: Ref: 6. Null_counts_customer_code

```
/*The purpose of this query is to get null_counts in customer_code c
olumn
```

```
for both gdb041.fact_sales_monthly and gdb041.dim_customer table */

SELECT
    'fact_table' as table_names,
    count(*) as null_counts
FROM
    gdb041.fact_sales_monthly
WHERE customer_code is null

UNION ALL

SELECT
    'customer_table' as table_names,
    count(*) as null_counts
FROM
    gdb041.dim_customer
WHERE customer_code is null
```

Decision:

- No null values detected

▼ **Confirm Referential Integrity**

What Was Checked?

- Whether all `customer_code` values in *fact_sales_monthly* exist in *dim_customer*

How it was Checked?

- Referential integrity for `customer_code`: **Ref: 7. Referential_integrity_for_customer_code**

```
-- This query helps to define whether Referential Integrity is intac
t or not--
-- if this query return no rows, then Referential Integrity is intac
t --

SELECT DISTINCT
    f.customer_code
FROM gdb041.fact_sales_monthly f
LEFT JOIN gdb041.dim_customer c
    ON f.customer_code = c.customer_code
WHERE c.customer_code IS NULL;

-- No rows return, referential integrity is intect --
```

Findings:

- All `customer_code` in *fact_sales_monthly* exist in *dim_customer*

Decision

- The referential integrity between the fact and dimension tables is intact.
- I can remove dependent descriptive columns (`customer, platform, channel`)
- I will not remove market column as I have seperate dimension table for market

— ***dim_product*** **will be regarded as DP —**

▼ **DP_Sample Data Check:**

What Was Checked?

- Row Counts
- Available Columns
- Grain
- Data type for all columns

How it was Checked?

SQL: **Ref: 8. DP_Sample_data_check**

▼ Code

```
/*
How many rows dim_product have?
*/
SELECT count(*) FROM gdb041.dim_product

/*
Check sample data
*/
SELECT * FROM gdb041.dim_product LIMIT 10

/*
The following query will help to understand to find the grain.
What each row in this dataset actually represents.
I am assuming that product_code is the grain.
If the query return no rows. The assumption is validated.
*/
SELECT
    count(*) as row_counts,
    product_code
FROM
    gdb041.dim_product
GROUP BY
    product_code
HAVING
    ROW_COUNTS>1

-- No rows return --
```

```
-- The grain is product_code --
 -- Check data type for all columns --
DESCRIBE dim_product
```

Findings:

- **Rows:** 397 rows
- **Columns:** 6 columns
- **Column List:** `product_code,division,segment,category,product,variant`
- **Grain:** No duplicate records were found at the defined grain `product_code`.
- **Data type:** All column is varchar(255)

Decisions:

- Each row in dim_product represents details for a specific product_code

Notes:

- If referential integrity between FSM and DP is intact, descriptive columns (`division, category, product`) can be removed.

▼ **Check referential integrity for product_code between FSM and DP**

Checklist:

- **Grain:** FSM: `date+product_code+customer_code`, DP: `product_code`
- **Data Type:** VARCHAR for both
- `product_code` is unique in *dim_product*

▼ **Whitespace_case_inconsistencies check for product_code in FSM and DP**

How it was checked?

SQL: Ref: 9. product_code_Whitespace_case_inconsistencies

```
/* The purpose of this query is to find whitespaces and case inconsi
stencies in the product_code columns*/
SELECT
    'fact_table' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT product_code) AS examples
FROM gdb041.fact_sales_monthly
WHERE product_code != UPPER(TRIM(product_code))

UNION ALL

SELECT
    'product_table' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT product_code) AS examples
```

```
FROM gdb041.dim_product
WHERE product_code != UPPER(TRIM(product_code));
```

Decision:

- No whitespace and case_inconsistencies detected

▼ **Null values for product_Code in FSM and DP**

How it was checked?

SQL: Ref: 10. Null_counts_product_code

```
/*The purpose of this query is to get null_counts in product_code co
lumn
for both gdb041.fact_sales_monthly and gdb041.dim_product table */

SELECT
    'fact_table' as table_names,
    count(*) as null_counts
FROM
    gdb041.fact_sales_monthly
WHERE product_code is null

UNION ALL

SELECT
    'product_table' as table_names,
    count(*) as null_counts
FROM
    gdb041.dim_product
WHERE product_code is null
```

Decision:

- No null values detected

▼ **Confirm Referential Integrity**

What Was Checked?

- Whether all `product_code` values in *fact_sales_monthly* exist in *dim_product*

How it was Checked?

- Referential integrity for `product_code`: **Ref: 11. Referential_integrity_for_product_code**

```
-- This query helps to define whether Referential Integrity is intac
t or not--
-- if this query return no rows, then Referential Integrity is intac
t --
SELECT
```

```
    fsm.product_code
FROM fact_sales_monthly fsm
LEFT JOIN dim_product dp
on fsm.product_code=dp.product_code
where dp.product_code is null

-- no rows return --
```

Findings:

- All `product_code` in *fact_sales_monthly* exist in *dim_product*

Decision

- The referential integrity between the fact and dimension tables is intact.
- I can remove dependent descriptive columns (`division, category, product`)

— dim_market will be regarded as DM —

▼ **DM_Sample Data Check:**

What Was Checked?

- Row Counts
- Available Columns
- Grain
- Data type for all columns

How it was Checked?

SQL: **Ref: 12. DM_Sample_data_check**

▼ Code

```
/*
How many rows dim_market have?
*/
SELECT count(*) FROM gdb041.dim_market

/*
Check sample data
*/
SELECT * FROM gdb041.dim_market LIMIT 5

/*
The following query will help to understand to find the grain.
What each row in this dataset actually represents.
I am assuming that market is the grain.
If the query return no rows. The assumption is validated.
*/
SELECT
```

```
    count(*) as row_counts,
    market
FROM
    gdb041.dim_market
GROUP BY
    market
HAVING
    ROW_COUNTS>1

-- No rows return --
-- The grain is market --

 -- Check data type for all columns --
DESCRIBE dim_market
```

Findings:

- **Rows:** 27 rows
- **Columns:** 6 columns
- **Column List:** `market, sub_zone, region`
- **Grain:** No duplicate records were found at the defined grain `market.`
- **Data type:** All column is varchar(255)

Decisions:

- Each row in dim_market represents details for a specific `market`

▼ **Check referential integrity for market between FSM and DM**

Checklist:

- **Grain:** FSM: `date+product_code+customer_code`, DP: `market`
- **Data Type:** VARCHAR for both
- `market` is unique in *dim_market*

▼ **Whitespace_case_inconsistencies check for market in FSM and DM**

How it was checked?

SQL: Ref: 13. market_Whitespace_case_inconsistencies

```
/* The purpose of this query is to find whitespaces and case inconsi
stencies in the market columns*/
SELECT
    'fact_table' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT market) AS examples
FROM gdb041.fact_sales_monthly
WHERE market != UPPER(TRIM(market))
```

```
UNION ALL

SELECT
    'market' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT market) AS examples
FROM gdb041.dim_market
WHERE market != UPPER(TRIM(market));
```

Decision:

- No whitespace and case_inconsistencies detected

▼ **Null values for market in FSM and DM**

How it was checked?

SQL: Ref: 14. Null_counts_market

```
/*The purpose of this query is to get null_counts in market column
for both gdb041.fact_sales_monthly and gdb041.market table */

SELECT
    'fact_table' as table_names,
    count(*) as null_counts
FROM
    gdb041.fact_sales_monthly
WHERE market is null

UNION ALL

SELECT
    'product_table' as table_names,
    count(*) as null_counts
FROM
    gdb041.market
WHERE market is null
```

Decision:

- No null values detected

▼ **Confirm Referential Integrity**

What Was Checked?

- Whether all `market` values in *fact_sales_monthly* exist in *dim_market*

How it was Checked?

- Referential integrity for `market`: **Ref: 15. Referential_integrity_for_market**

```
-- This query helps to define whether Referential Integrity is intac
t or not--
-- if this query return no rows, then Referential Integrity is intac
t --
SELECT
    fsm.market
FROM fact_sales_monthly fsm
LEFT JOIN dim_market dp
on fsm.market=dp.market
where dp.market is null

-- no rows return --
```

Findings:

- All `market` in *fact_sales_monthly* exist in *dim_product*

Decision

- The referential integrity between the fact and dimension tables is intact.

— Remove `customer, platform, channel, division, category, product` from fact_sales_monthly —

▼ **Create a view fact_sales_monthly_cleaned without selected rows**

Ref: 16. vw_fact_sales_monthly_cleaned

```
CREATE or REPLACE VIEW fact_sales_monthly_cleaned as
SELECT
    `date`,
    product_code,
    customer_code,
    market,
    sold_quantity
FROM
    fact_sales_monthly
```

— *fact_sales_monthly_cleaned* will be regarded as FSMC —

▼ **Date Analysis from FSMC:**

What Was Checked?

- Date is daily, monthly, weekly or yearly?
- Start Date, End Date
- Expected Month Count, Actual Month Count
- Missing Month
- Date consistency

How it was Checked?

▼ Date Interval: SQL→ **Ref: 17. vw_FSMC_data_interval**

```
-- Get all unique dates in chronological order
SELECT DISTINCT `date`
FROM gdb041.fact_sales_monthly_cleaned
ORDER BY `date`;

-- Find the gap (in days) between each consecutive date
WITH ordered_dates AS (
    SELECT DISTINCT `date`
    FROM gdb041.fact_sales_monthly_cleaned
    order by `date`
),
date_gaps as (
    SELECT
        `date` as curr_date,
        LAG(`date`) OVER (ORDER BY `date`) as previous_date,
        DATEDIFF(`date`, LAG(`date`) OVER (ORDER BY `date`)) as days
_gap
    FROM ordered_dates
)
SELECT
    days_gap,
    COUNT(*) as frequency,
    MIN(curr_date) as example_date
FROM date_gaps
WHERE days_gap IS NOT NULL
GROUP BY days_gap
ORDER BY frequency DESC;

-- I can confirm that the date is monthly--
```

▼ Start/End Date, Expected/Actual Month Count, Missing Month: **Ref: 18. vw_FSMC_Verify_missing_months**

```
-- I will verify actual_date_count with date count from (min date to
max date)--
-- it will help to validate if there is any missing month--
SELECT
    MIN(date) as start_date,
    MAX(date) as end_date,
    PERIOD_DIFF(
        DATE_FORMAT(MAX(date), '%Y%m'),
        DATE_FORMAT(MIN(date), '%Y%m')
    ) + 1 as expected_month_count,
    COUNT(DISTINCT date) as actual_date_count
FROM gdb041.fact_sales_monthly_cleaned;
```

```
-- There is no missing months --
```

▼ Date Consistency: **Ref: 19. vw_FSMC_Verify_date_consistency**

```
-- Are all the dates on the same day of month? (e.g., always 1st or
last day)
SELECT
    DAY(date) as day_of_month,
    COUNT(*) as frequency,
    MIN(date) as example_date
FROM gdb041.fact_sales_monthly_cleaned
GROUP BY day_of_month
ORDER BY frequency DESC;

-- all the dates starts from 1st day of the month--
```

Findings:

- Date interval is monthly.
- Start Date: 2017-09-01
- End Date: 2021-12-01
- Expected Month count: 52
- Actual month count: 52
- No missing months found
- All the dates starts from 1st day of the month

Decision

- The date column is suitable for analysis at a monthly granularity.

▼ **Check Null values in FSMC:**

How?

SQL: **Ref: 20. vw_FSMC_null_values**

```
/*This query helps to number of rows where the value is null for each c
olumn*/

SELECT
    COUNT(*) AS total_rows,
    SUM(CASE WHEN `date` IS NULL THEN 1 ELSE 0 END) AS null_date,
    SUM(CASE WHEN product_code IS NULL THEN 1 ELSE 0 END) AS null_produ
ct_code,
    SUM(CASE WHEN customer_code IS NULL THEN 1 ELSE 0 END) AS null_cust
```

```
omer_code,
    SUM(CASE WHEN market IS NULL THEN 1 ELSE 0 END) AS null_market,
    SUM(CASE WHEN sold_quantity IS NULL THEN 1 ELSE 0 END) AS null_sold
_quantity
FROM fact_sales_monthly_cleaned

-- No null cases found --
```

Findings:

- No null cases found

▼ **Check negative and zero values in sold_quantity:**

How?

SQL: **Ref: 21. vw_FSMC_SC_ngtv_zero**

```
/*This query helps to get number of rows where the value is
either negative or zero in sold_quantity column*/

SELECT
    count(*) as rows_counts,
    SUM(CASE WHEN sold_quantity<0 then 1 else 0 END) as negative_count
s,
    SUM(CASE WHEN sold_quantity=0 then 1 else 0 END) as zero_counts
FROM
    fact_sales_monthly_cleaned

-- 783 rows found zero --
```

Findings:

- 783 rows found zero.
- No negative values found

Where sold_quantity is ZERO?

- **How?**
  - SQL: Ref: **Ref: 22. vw_FSMC_SC_zero**

```
-- Sold_quantity by date
SELECT
    `date`,
    COUNT(*) AS zero_qty_count
FROM fact_sales_monthly_cleaned
WHERE sold_quantity = 0
GROUP BY `date`

-- only 1 date found 01-09-2017 --
```

```
-- Sold_quantity by product_code
SELECT
    product_code,
    COUNT(*) AS zero_qty_count
FROM fact_sales_monthly_cleaned
WHERE sold_quantity = 0
GROUP BY product_code

-- 87 product_code found--

-- Sold_quantity by customer_code
SELECT
    customer_code,
    COUNT(*) AS zero_qty_count
FROM fact_sales_monthly_cleaned
WHERE sold_quantity = 0
GROUP BY customer_code

-- 9 customer_code found having 87 rows each--

-- Sold_quantity by product_code
SELECT
    market,
    COUNT(*) AS zero_qty_count
FROM fact_sales_monthly_cleaned
WHERE sold_quantity = 0
GROUP BY market

-- only germany--
```

Findings:

- Sold_quantity was zero for 9 customers on 01-09-2017 only in Germany.
- I assumed that no products were sold on this day for those customers in germany.

— JOIN FSMC and Gross_sales to get gross_price (GP) —

▼ GP_Sample Data Check:

What Was Checked?

- Row Counts
- Available Columns
- Grain
- Data type for all columns

How it was Checked?

SQL: **Ref: 23. GP_Sample_data_check**

▼ Code

```
/*
How many rows gross_price have?
*/
SELECT count(*) FROM gdb056.gross_price

/*
Check sample data
*/
SELECT * FROM gdb056.gross_price LIMIT 10

/*
The following query will help to understand to find the grain.
What each row in this dataset actually represents.
Assume product_code as grain first.
If the query return no rows. The assumption is validated.
*/
SELECT
    count(*) as row_counts,
    product_code
FROM
    gdb056.gross_price
GROUP BY
    product_code
HAVING
    ROW_COUNTS>1

-- multiple rows return --
-- check product_code+fiscal_year --
SELECT
    count(*) as row_counts,
    product_code,
    fiscal_year
FROM
    gdb056.gross_price
GROUP BY
    product_code, fiscal_year
HAVING
    ROW_COUNTS>1

-- No rows return --
-- the grain is prodcut_code+fiscal_year --

 -- Check data type for all columns --
DESCRIBE gdb056.gross_price
```

Findings:

- **Rows:** 1197 rows
- **Columns: 3** columns
- **Column List:** `product_code,fiscal_year,gross_price`
- **Grain:** No duplicate records were found at the defined grain `prodcut_code+fiscal_year.`
- **Data type:** `product_code` varchar(255), `fiscal_year` int, `gross_price` decimal (15,10)

Decisions:

- Each row in *gdb056.gross_price* represents details for a specific `product_code+fiscal_year`

Notes:

- I have to add fiscal_year column in FSMC before join FSMC and GP

▼ **Add fiscal_year column in FSMC**

Fiscal_year starts from September in Atliq Hardware.

SQL: **Ref: 24. vw_fact_sales_FY**

```
/*This row will create a new view from fact_sales_monthly_cleaned with
the new column fiscal_year derived from date column*/

CREATE or REPLACE VIEW vw_fact_sales_FY as
SELECT
    `date`,
    CASE
        WHEN month(`date`)<09 THEN year(`date`) else
        year(`date`)+1
    END as fiscal_year,
    product_code,
    customer_code,
    market,
    sold_quantity
FROM
    fact_sales_monthly_cleaned

-- check data type of new column --
DESCRIBE vw_fact_sales_fy fiscal_year
```

Notes:

- A new column fiscal_year is added
- Data type for fiscal_year is int

— vw_fact_sales_FY will be regarded as FSFY —

▼ **Join FSFY and Gross_price**

Checklist:

- **Grain:** FSFY: `date+product_code+customer_code`, GP: `product_code+fiscal_year`
- The join will be based on `product_code+fiscal_year`
- **Data Type:** varchar for `product_code` in both dataset, int for `fiscal_year` in both dataset.

▼ **Whitespace_case_inconsistencies check for product_code in GP**

How it was checked?

SQL: **Ref: 25. GP_witespce_inconsistencies**

```
/* The purpose of this query is to find whitespaces and case inconsi
stencies in the product_code columns*/
SELECT
    'GP' AS table_name,
    COUNT(*) AS non_standard_rows,
    GROUP_CONCAT(DISTINCT product_code) AS examples
FROM gdb056.gross_price
WHERE product_code != UPPER(TRIM(product_code));
```

Decision:

- No whitespace and case_inconsistencies detected

▼ **Null values for product_code and fiscal_year in FSFY and GP**

How it was checked?

SQL: **Ref: 26. vw_null_counts_FSFY_GP**

```
/*The purpose of this query is to get null_counts in product_code an
d fiscal_year column
for both vw_fact_sales_fy and gdb056.gross_price */

SELECT
    'fact_table' as table_names,
    SUM(CASE WHEN product_code is null THEN 1 else 0 END) as null_pr
oduct_code,
    SUM(CASE WHEN fiscal_year is null THEN 1 else 0 END) as null_fis
cal_year
FROM
    vw_fact_sales_fy

UNION ALL

SELECT
    'GP' as table_names,
    SUM(CASE WHEN product_code is null THEN 1 else 0 END) as null_pr
oduct_code,
    SUM(CASE WHEN fiscal_year is null THEN 1 else 0 END) as null_fis
cal_year
```

```
FROM
    gdb056.gross_price
```

Decision:

- No null values detected

▼ **Confirm Referential Integrity**

What Was Checked?

- Whether all `product_code and fiscal_year` values in *vw_fact_sales_fy* exist in *gross_price*

How it was Checked?

- **Ref: 27. Referential_integrity_check_for_FSFY_GP**

```
-- This query helps to define whether Referential Integrity is intac
t or not--
-- if this query return no rows, then Referential Integrity is intac
t --

SELECT
    FSFY.product_code,
    FSFY.fiscal_year
FROM
    vw_fact_sales_fy FSFY
LEFT JOIN
    gdb056.gross_price GP
    on FSFY.product_code=GP.product_code
    AND FSFY.fiscal_year=GP.fiscal_year
WHERE GP.product_code is null

-- no rows return --
```

Findings:

- All all `product_code and fiscal_year` values in *fact_sales_monthly* exist in *gross_price*

▼ **Join FSFY and Gross_price**

Ref: 28. vw_fact_sales_gp

```
Create or Replace View vw_fact_sales_GP as

SELECT
    `date`,
    FSFY.fiscal_year,
    FSFY.product_code,
    FSFY.customer_code,
    FSFY.market,
    FSFY.sold_quantity,
```

```
    GP.gross_price
FROM
    vw_fact_sales_fy FSFY
LEFT JOIN
    gdb056.gross_price GP
    on FSFY.product_code=GP.product_code
    AND FSFY.fiscal_year=GP.fiscal_year
```

▼ **Verify Joining**

What was checked?

- Row counts
- Total sold_quantity
- gross_price null

How?

SQL: REF: **29. Verify_join_FSFY_GP**

```
/* The following queries helpe to verify whether joining was success
ful */

-- Check row counts between vw_fact_sales_fy and vw_fact_sales_gp --
 SELECT
    (SELECT count(*) FROM vw_fact_sales_fy) as original_count,
    count(*) as join_count
FROM
    vw_fact_sales_gp

-- row counts same --

-- Check total_sum of sold_quantity between vw_fact_sales_fy and vw_
fact_sales_gp --
SELECT
    (SELECT sum(sold_quantity) from vw_fact_sales_fy) as original_to
tal,
    sum(sold_quantity) as join_total
FROM
    vw_fact_sales_gp

-- Count how many rows have NULL gross_price --
SELECT COUNT(*) AS missing_gross_price
FROM vw_fact_sales_gp
WHERE gross_price IS NULL;
```

Findings:

- Row counts same, no issue
- Total sold_quantity same, no issue
- in `gross_price` column, no null detected.

Execution Pending:

-