AtliQ Hardware Data Analysis Project

Introduction

AtliQ Hardware is a global company that sells hardware products through various channels across multiple regions. This project aims to analyze their sales data to extract meaningful business insights, with a special focus on the Indian market.

Database Schema Overview
The AtliQ Hardware database consists of the following tables:

dim_customer: Customer information including customer code, name, platform, channel, market, sub-zone, and region
dim_date: Date information including calendar date and fiscal year
dim_product: Product information including product code, division, segment, category, product name, and variant
fact_actual_est: Actual and estimated sales data
fact_forecast_monthly: Monthly forecast data
fact_freight_cost: Freight and other cost percentages by market and fiscal year
fact_gross_price: Gross price information by product and fiscal year
fact_manufacturing_cost: Manufacturing cost information by product and year
fact_post_invoice_deductions: Post-invoice deduction information
fact_pre_invoice_deductions: Pre-invoice deduction information
fact_sales_monthly: Monthly sales data

Data Cleaning and Quality Assessment

```
-- Check for NULL values in dim_customer
SELECT *
FROM dim_customer
WHERE customer_code IS NULL
OR customer IS NULL
OR platform IS NULL
OR channel IS NULL
OR market IS NULL
OR sub_zone IS NULL
OR region IS NULL;

-- Handling NULL Values in dim_customer
SELECT customer_code,
COALESCE(customer, 'N/A') AS customer,
COALESCE(platform, 'N/A') AS platform,
COALESCE(channel, 'N/A') AS channel,
COALESCE(market, 'N/A') AS market,
COALESCE(sub_zone, 'N/A') AS sub_zone,
COALESCE(region, 'N/A') AS region
FROM dim_customer;

-- Check for duplicates in dim_customer
SELECT customer_code, customer, COUNT(*)
FROM dim_customer
GROUP BY customer_code, customer
HAVING COUNT(*) > 1;
```

```sql
-- Check for NULL values in dim_product
SELECT *
FROM dim_product
WHERE product_code IS NULL
OR division IS NULL
OR segment IS NULL
OR category IS NULL
OR product IS NULL
OR variant IS NULL;
```

```sql
-- Handling NULL Values in dim_product
SELECT product_code,
COALESCE(division, 'N/A') AS division,
COALESCE(segment, 'N/A') AS segment,
COALESCE(category, 'N/A') AS category,
COALESCE(product, 'N/A') AS product,
COALESCE(variant, 'N/A') AS variant
FROM dim_product;
```

```sql
-- Check for duplicates in dim_product
SELECT product_code, product, COUNT(*)
FROM dim_product
GROUP BY product_code, product
HAVING COUNT(*) > 1;
```

```sql
-- Check for NULL values in fact_sales_monthly
SELECT *
FROM fact_sales_monthly
WHERE date IS NULL
OR fiscal_year IS NULL
OR product_code IS NULL
OR customer_code IS NULL
OR sold_quantity IS NULL;
```

```sql
-- Check for duplicates in fact_sales_monthly
SELECT date, fiscal_year, product_code, customer_code, COUNT(*)
FROM fact_sales_monthly
GROUP BY date, fiscal_year, product_code, customer_code
HAVING COUNT(*) > 1;
```

Insights: The data quality assessment revealed no NULL values in critical fields across customer, product, and sales tables. No duplicates were found in customer or product tables, indicating reliable data for analysis. These checks ensure the integrity of subsequent analyses.

Financial Analytics

```sql
-- Create a function to get fiscal year by passing the date
CREATE FUNCTION `get_fiscal_year`(calendar_date DATE)
RETURNS int
DETERMINISTIC
BEGIN
DECLARE fiscal_year INT;
```

```sql
SET fiscal_year = YEAR(DATE_ADD(calendar_date, INTERVAL 4 MONTH));
RETURN fiscal_year;
END
```

-- Get all the sales transaction data for Croma India (customer_code: 90002002) in fiscal year 2021

```sql
SELECT * FROM fact_sales_monthly
WHERE
customer_code=90002002 AND
get_fiscal_year(date)=2021
ORDER BY date asc
LIMIT 100000;
```

-- Perform joins to pull product information

```sql
SELECT s.date, s.product_code, p.product, p.variant, s.sold_quantity
FROM fact_sales_monthly s
JOIN dim_product p
ON s.product_code=p.product_code
WHERE
customer_code=90002002 AND
get_fiscal_year(date)=2021
LIMIT 1000000;
```

-- Performing join with 'fact_gross_price' table and generating required fields

```sql
SELECT
s.date,
s.product_code,
p.product,
p.variant,
s.sold_quantity,
g.gross_price,
ROUND(s.sold_quantity*g.gross_price,2) as gross_price_total
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=get_fiscal_year(s.date)
AND g.product_code=s.product_code
WHERE
customer_code=90002002 AND
get_fiscal_year(s.date)=2021
LIMIT 1000000;
```

-- Generate monthly gross sales report for Croma India for all the years

```sql
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=90002002
GROUP BY date;
```

-- Generate monthly gross sales report for any customer using stored procedure

```
CREATE PROCEDURE `get_monthly_gross_sales_for_customer`(
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
```

-- Write a stored proc that can retrieve market badge

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

# Default market is India

```
IF in_market = "" THEN
SET in_market="India";
END IF;
```

# Retrieve total sold quantity for a given market in a given 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;
```

# Determine Gold vs Silver status

```
IF qty > 5000000 THEN
SET out_level = 'Gold';
ELSE
SET out_level = 'Silver';
END IF;
END
```

-- Generate a yearly report for Croma India with fiscal year and total gross sales

```
select
get_fiscal_year(date) as fiscal_year,
sum(round(sold_quantity*g.gross_price,2)) as yearly_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=90002002
group by get_fiscal_year(date)
order by fiscal_year;
```

Insights: The financial analytics queries provide comprehensive insights into sales performance. The fiscal year function enables accurate financial reporting aligned with AtliQ Hardware's fiscal calendar (starting in May). The detailed transaction analysis for Croma India reveals specific product performance and revenue contributions. The stored procedures allow flexible analysis across different customers and markets, with a special focus on India as the default market. The yearly report for Croma India shows growth trends over time, helping identify patterns in this key Indian retailer's performance.

Customer and Product Analytics

```
-- Include pre-invoice deductions in Croma detailed report
SELECT
s.date,
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=get_fiscal_year(s.date)
AND g.product_code=s.product_code
JOIN fact_pre_invoice_deductions  pre
ON pre.customer_code = s.customer_code AND
pre.fiscal_year=get_fiscal_year(s.date)
WHERE
s.customer_code=90002002 AND
get_fiscal_year(s.date)=2021
LIMIT 1000000;
```

```
-- Same report but for all customers in 2021
SELECT
s.date,
s.product_code,
p.product,
p.variant,
s.sold_quantity,
g.gross_price as gross_price_per_item,
```

```sql
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=get_fiscal_year(s.date)
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=get_fiscal_year(s.date)
WHERE
get_fiscal_year(s.date)=2021
LIMIT 1000000;
```

-- Creating dim_date and joining with this table to avoid using the function 'get_fiscal_year()'

```sql
SELECT
s.date,
dt.fiscal_year,
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_date dt
ON dt.calendar_date = s.date
JOIN dim_product p
ON s.product_code=p.product_code
JOIN fact_gross_price g
ON g.fiscal_year=dt.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=dt.fiscal_year
WHERE
dt.fiscal_year=2021
LIMIT 1500000;
```

-- Added the fiscal year in the fact_sales_monthly table itself

```sql
SELECT
s.date,
s.fiscal_year,
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;
```

-- Get the net_invoice_sales amount using the CTE's

```
WITH cte1 AS (
        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
)
SELECT
*,
(1-pre_invoice_discount_pct) *gross_price_total as net_invoice_sales
FROM cte1
LIMIT 1500000;
```

-- Creating the view `sales_preinv_discount`

```
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,
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;
```

-- Now generate net_invoice_sales using the above created view

```
SELECT
*,
(gross_price_total-pre_invoice_discount_pct*gross_price_total) as net_invoice_sales
FROM sales_preinv_discount;
```

-- Create a view for post invoice deductions: `sales_postinv_discount`

```
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_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;
```

-- Create a report for net sales

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

-- Finally creating the view `net_sales`

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

-- Get top 5 market by net sales in fiscal year 2021

```
SELECT
market,
round(sum(net_sales)/1000000,2) as net_sales_mln
```

```sql
FROM net_sales
where fiscal_year=2021
group by market
order by net_sales_mln desc
limit 5;

-- Stored proc to get top n markets by net sales for a given year
CREATE PROCEDURE `get_top_n_markets_by_net_sales`(
IN in_fiscal_year INT,
IN in_top_n INT
) DETERMINISTIC
BEGIN
SELECT
market,
round(sum(net_sales)/1000000,2) as net_sales_mln
FROM net_sales
where fiscal_year=in_fiscal_year
group by market
order by net_sales_mln desc
limit in_top_n;
END;

-- stored procedure that takes market, fiscal_year and top n as input and returns top n customers by net sales
CREATE PROCEDURE `get_top_n_customers_by_net_sales`(
IN in_market VARCHAR(45),
IN in_fiscal_year INT,
IN in_top_n INT
) DETERMINISTIC
BEGIN
select
c.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;

-- Find customer wise net sales percentage contribution
with cte1 as (
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=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;
```

-- Find customer wise net sales distribution per region for FY 2021

```
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;
```

-- Find out top 3 products from each division by total quantity sold in a given year

```
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),
cte2 as
(select
*,
dense_rank() over (partition by division order by total_qty desc) as drnk
from cte1)
select * from cte2 where drnk<=3;
```

-- Creating stored procedure for the above query

```
CREATE PROCEDURE `get_top_n_products_per_division_by_qty_sold`(
IN in_fiscal_year INT,
IN in_top_n INT
) DETERMINISTIC
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;
```

-- Create a view for gross sales

```
CREATE VIEW `gross_sales` 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
from fact_sales_monthly s
join dim_product p
on s.product_code=p.product_code
join dim_customer c
on s.customer_code=c.customer_code
join fact_gross_price g
on g.fiscal_year=s.fiscal_year
and g.product_code=s.product_code;
```

-- Write a stored procedure to get the top n products by net sales for a given year

```
CREATE PROCEDURE get_top_n_products_by_net_sales(
IN in_fiscal_year int,
IN in_top_n int
)
BEGIN
select
p.product,
round(sum(net_sales)/1000000,2) as net_sales_mln
from net_sales s
join dim_product p ON s.product_code=p.product_code
where fiscal_year=in_fiscal_year
group by p.product
order by net_sales_mln desc
limit in_top_n;
END;
```

-- Write a query to get region wise % net sales breakdown by customers

```
with cte1 as(
SELECT
c.customer,
c.region,
round(sum(s.net_sales)/1000000,2) as net_sales_mln
FROM net_sales s
JOIN dim_customer c ON s.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
from cte1
order by region, net_sales_mln DESC;
```

-- Retrieve the top 2 markets in every region by their gross sales amount in FY=2021

```
with cte1 as (
select
c.market,
c.region,
round(sum(gross_price_total)/1000000,2) as gross_sales_mln
from gross_sales s
join dim_customer c
on c.customer_code=s.customer_code
where fiscal_year=2021
group by market, region
order by gross_sales_mln desc
),
cte2 as (
select *,
dense_rank() over(partition by region order by gross_sales_mln desc) as drnk
from cte1
)
select * from cte2 where drnk<=2;
```

Insights: The customer and product analytics provide deep insights into sales performance, discount impacts, and ranking of products and customers. The views created simplify complex calculations for net sales, while window functions enable percentage contribution analysis. The stored procedures offer flexibility to analyze top performers across different dimensions. These analyses reveal which products drive sales in each division and how customers contribute to regional performance, with particular relevance for understanding the Indian market within the APAC region.

Supply Chain Analytics

-- Create fact_act_est table

```
drop table if exists fact_act_est;

create table fact_act_est
(
select
s.date as date,
s.fiscal_year as fiscal_year,
s.product_code as product_code,
s.customer_code as customer_code,
s.sold_quantity as sold_quantity,
f.forecast_quantity as forecast_quantity
from
fact_sales_monthly s
left join fact_forecast_monthly f
using (date, customer_code, product_code)
)
union
(
select
f.date as date,
f.fiscal_year as fiscal_year,
f.product_code as product_code,
f.customer_code as customer_code,
s.sold_quantity as sold_quantity,
f.forecast_quantity as forecast_quantity
from
fact_forecast_monthly f
left join fact_sales_monthly s
using (date, customer_code, product_code)
);

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;
```

-- create the trigger to automatically insert record in fact_act_est table whenever insertion happens in

```
fact_sales_monthly
CREATE DEFINER=CURRENT_USER TRIGGER `fact_sales_monthly_AFTER_INSERT` AFTER 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;
```

-- create the trigger to automatically insert record in fact_act_est table whenever insertion happens in

```
fact_forecast_monthly
CREATE DEFINER=CURRENT_USER TRIGGER `fact_forecast_monthly_AFTER_INSERT` AFTER 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;
```

-- To see all the Triggers

```
show triggers;
```

-- Creating the table "session_logs" and inserting records

```
CREATE TABLE session_logs (`ts` DATETIME, `session_id` INT, `user_id` INT, `log` TEXT);
INSERT INTO `session_logs`
(`ts`, `session_id`, `user_id`, `log`)
VALUES
('2022-10-04 08:14:07', '898812', '523', 'CLICKED | Courses Button'),
('2022-10-14 08:18:35', '898812', '523', 'NAVIGATE BACK | Python course page , codebasics.io'),
('2022-10-16 12:07:00', '965345', '523', 'REVIEW GENERATED | Data analytics in power bi'),
('2022-10-22 14:09:22', '188567', '707', 'NEW LOGIN | New login, user name: tasty@jalebi.com'),
('2022-10-22 18:10:06', '188567', '707', 'COURSE PURCHASED | Data analytics in power bi, user name:
tasty@jalebi.com');
```

-- Delete logs that are less than 5 days old

```
delimiter |
CREATE EVENT e_daily_log_purge
ON SCHEDULE
EVERY 5 SECOND
COMMENT 'Purge logs that are more than 5 days old'
DO
BEGIN
delete from session_logs
where DATE(ts) < DATE("2022-10-22") - interval 5 day;
END |
delimiter ;
```

-- drop the event

```
drop event if exists e_daily_log_purge;
```

-- Forecast accuracy report using cte

```
SET SESSION sql_mode = '';
with forecast_err_table as (
select
a.customer_code,
sum(sold_quantity),
sum(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_actual_est a
where a.fiscal_year = 2021
group by a.customer_code)
select
e.*,
c.customer as customer_name,
c.market,
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;
```

-- Write a stored proc for the same

```
CREATE PROCEDURE `get_forecast_accuracy`(
IN in_fiscal_year INT
)
BEGIN
SET SESSION sql_mode = '';
with forecast_err_table as (
select
a.customer_code,
sum(sold_quantity),
sum(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_actual_est a
where a.fiscal_year = 2021
group by a.customer_code)
select
e.*,
c.customer as customer_name,
c.market,
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;
END;
```

-- Forecast accuracy report using temporary table

```
ALTER TABLE fact_act_est
MODIFY forecast_quantity BIGINT SIGNED NOT NULL,
MODIFY sold_quantity BIGINT SIGNED NOT NULL;
drop table if exists forecast_err_table;
create temporary table forecast_err_table
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,
sum(s.forecast_quantity-s.sold_quantity)*100/sum(s.forecast_quantity) as net_error_pct,
sum(abs(s.forecast_quantity-s.sold_quantity)) as abs_error,
sum(abs(s.forecast_quantity-sold_quantity))*100/sum(s.forecast_quantity) as abs_error_pct
from fact_act_est s
join dim_customer c
on s.customer_code = c.customer_code
where s.fiscal_year=2021
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;
```

-- Create a new user 'thor'

```
create user 'thor'@'localhost' identified by 'thor';
```

-- Allow certain access to 'thor' user for the database

```
grant select on dim_customer to 'thor'@'localhost';
grant select on dim_product to 'thor'@'localhost';
grant execute on procedure get_forecast_accuracy to 'thor'@'localhost';
```

-- See all the access for 'thor' user

```
show grants for 'thor'@'localhost';
```

-- Write a query for customers whose forecast accuracy has dropped from 2020 to 2021
# step 1: Get forecast accuracy of FY 2021 and store that in a temporary table

```
drop table if exists forecast_accuracy_2021;
create temporary table forecast_accuracy_2021
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=2021
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;
```

# step 2: Get forecast accuracy of FY 2020 and store that also in a temporary table

```
drop table if exists forecast_accuracy_2020;
create temporary table forecast_accuracy_2020
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=2020
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;
```

# step 3: Join forecast accuracy tables for 2020 and 2021 using a customer_code

```
select
f_2020.customer_code,
f_2020.customer_name,
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;
```

Insights: The supply chain analytics focus on forecast accuracy and data management. The helper table combines actual and forecast data for analysis, while triggers automate data maintenance. The forecast accuracy reports identify customers with reliable versus unreliable forecasts, which is crucial for inventory planning. The year-over-year comparison highlights where forecasting has deteriorated, enabling model improvements. User management and indexing examples demonstrate database optimization techniques that improve performance for large-scale supply chain analytics.

India Market Analysis (Focus Area)

-- Top-selling products in India

```
SELECT
dp.product AS Product_Name,
dp.category AS Category,
dp.segment AS Segment,
SUM(fsm.sold_quantity) AS Quantity_Sold
FROM fact_sales_monthly fsm
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN dim_customer dc ON fsm.customer_code = dc.customer_code
WHERE dc.market = 'India'
GROUP BY dp.product, dp.category, dp.segment
ORDER BY Quantity_Sold DESC
LIMIT 10;
```

-- Top product categories in India

```
SELECT
dp.category AS Category,
dp.division AS Division,
SUM(fsm.sold_quantity) AS Quantity_Sold
FROM fact_sales_monthly fsm
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN dim_customer dc ON fsm.customer_code = dc.customer_code
WHERE dc.market = 'India'
GROUP BY dp.category, dp.division
ORDER BY Quantity_Sold DESC
LIMIT 10;
```

-- Customer breakdown in India by platform and channel

```
SELECT
platform,
channel,
COUNT(*) AS customer_count
FROM dim_customer
WHERE market = 'India'
GROUP BY platform, channel
ORDER BY customer_count DESC;
```

-- Customer breakdown in India by sub-zone

```
SELECT
sub_zone,
COUNT(*) AS customer_count
FROM dim_customer
WHERE market = 'India'
GROUP BY sub_zone
ORDER BY customer_count DESC;
```

-- Total Revenue by different customers in Sub-zone of India

```
SELECT dc.sub_zone,
SUM(fsm.sold_quantity * fgp.gross_price) AS Total_Revenue
FROM fact_sales_monthly fsm
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN dim_customer dc ON fsm.customer_code = dc.customer_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dc.sub_zone
ORDER BY Total_Revenue DESC;
```

```
-- Number of Sales by Sub-zone in India
SELECT dc.sub_zone,
SUM(fsm.sold_quantity) AS total_sales
FROM fact_sales_monthly fsm
JOIN dim_customer dc ON fsm.customer_code = dc.customer_code
WHERE dc.market = 'India'
GROUP BY dc.sub_zone
ORDER BY total_sales DESC;

-- Top 5 customers by total revenue in India
WITH Customer_wise_revenue_cte1 as(
SELECT
dc.customer_code,
dc.customer,
SUM(fsm.sold_quantity * fgp.gross_price) as Total_Revenue
FROM
dim_customer dc
INNER JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY
dc.customer_code, dc.customer
ORDER BY
Total_Revenue DESC)
SELECT
customer_code,
customer,
Total_Revenue
FROM
Customer_wise_revenue_cte1
LIMIT 5;

-- Frequency of Purchases in India
SELECT dc.customer_code, dc.customer,
COUNT(DISTINCT fsm.date) AS Purchase_Frequency
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
WHERE dc.market = 'India'
GROUP BY dc.customer_code, dc.customer
ORDER BY Purchase_Frequency DESC
LIMIT 15;

-- Calculate the average order value for each customer in India
SELECT dc.customer_code, dc.customer,
Round(AVG(fsm.sold_quantity * fgp.gross_price), 2) AS average_order_value
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dc.customer_code, dc.customer
ORDER BY average_order_value DESC
```

```
LIMIT 15;

-- Product category which was purchased on frequent basis in India
SELECT dp.category AS category, COUNT(*) AS purchase_frequency
FROM dim_product dp
JOIN fact_sales_monthly fsm ON dp.product_code = fsm.product_code
JOIN dim_customer dc ON fsm.customer_code = dc.customer_code
WHERE dc.market = 'India'
GROUP BY dp.category
ORDER BY purchase_frequency DESC
LIMIT 15;

-- Calculate the percentage of total sales contributed by each product category in India
WITH CategorySales AS (
SELECT dp.category AS category,
dp.division AS division,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_sales
FROM fact_sales_monthly fsm
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN dim_customer dc ON fsm.customer_code = dc.customer_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dp.category, dp.division
),
TotalSales AS (
SELECT SUM(total_sales) AS total_sales_amount
FROM CategorySales)
SELECT cs.category, cs.division, cs.total_sales,
(cs.total_sales / ts.total_sales_amount * 100) AS sales_percentage
FROM CategorySales cs
CROSS JOIN TotalSales ts
ORDER BY cs.total_sales DESC
LIMIT 15;

-- Analyze sales performance across different sub-zones in India
SELECT dc.sub_zone,
COUNT(DISTINCT dc.customer_code) AS customer_count,
SUM(fsm.sold_quantity) AS total_quantity_sold,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue,
ROUND(AVG(fsm.sold_quantity * fgp.gross_price), 2) AS avg_order_value
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dc.sub_zone
ORDER BY total_revenue DESC;

-- Top product categories by sub-zone in India
SELECT dc.sub_zone, dp.category,
SUM(fsm.sold_quantity) AS quantity_sold,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue
FROM dim_customer dc
```

```
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dc.sub_zone, dp.category
ORDER BY dc.sub_zone, total_revenue DESC;
```

Insights: The India market analysis provides comprehensive insights into this key market. The top-selling products and categories reveal what resonates most with Indian consumers. Customer breakdown shows a balanced distribution between Brick & Mortar and E-Commerce platforms, with Direct channel predominating. All customers are concentrated in the India sub-zone, indicating a focused geographic approach. The top customers by revenue represent strategic partnerships that deserve special attention. Purchase frequency and average order value analyses identify high-value customers, while category performance shows which product lines drive the business. These insights collectively inform tailored strategies for the Indian market.

Comparative Analysis

-- Compare India with other markets in terms of revenue

```
SELECT
CASE
   WHEN dc.market = 'India' THEN 'India'
   ELSE 'Other Markets'
END AS market_group,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue,
COUNT(DISTINCT dc.customer_code) AS customer_count,
ROUND(SUM(fsm.sold_quantity * fgp.gross_price) / COUNT(DISTINCT dc.customer_code), 2) AS
revenue_per_customer
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
GROUP BY market_group
ORDER BY total_revenue DESC;
```

-- Compare product category preferences between India and other markets

```
SELECT
dp.category,
CASE
   WHEN dc.market = 'India' THEN 'India'
   ELSE 'Other Markets'
END AS market_group,
SUM(fsm.sold_quantity) AS quantity_sold,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
GROUP BY dp.category, market_group
ORDER BY dp.category, total_revenue DESC;
```

-- Compare performance across all regions

```
SELECT dc.region,
```

```sql
COUNT(DISTINCT dc.customer_code) AS customer_count,
SUM(fsm.sold_quantity) AS total_quantity_sold,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue,
ROUND(AVG(fsm.sold_quantity * fgp.gross_price), 2) AS avg_order_value
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
GROUP BY dc.region
ORDER BY total_revenue DESC;

-- Top markets by region
WITH market_ranking AS (
SELECT
dc.region,
dc.market,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue,
RANK() OVER (PARTITION BY dc.region ORDER BY SUM(fsm.sold_quantity * fgp.gross_price) DESC) AS market_rank
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
GROUP BY dc.region, dc.market
)
SELECT region, market, total_revenue
FROM market_ranking
WHERE market_rank <= 3
ORDER BY region, market_rank;

-- Channel effectiveness by market
SELECT dc.market, dc.channel,
COUNT(DISTINCT dc.customer_code) AS customer_count,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue,
ROUND(SUM(fsm.sold_quantity * fgp.gross_price) / COUNT(DISTINCT dc.customer_code), 2) AS
revenue_per_customer
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
GROUP BY dc.market, dc.channel
ORDER BY dc.market, total_revenue DESC;

-- Platform effectiveness by market
SELECT dc.market, dc.platform,
COUNT(DISTINCT dc.customer_code) AS customer_count,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_revenue,
ROUND(SUM(fsm.sold_quantity * fgp.gross_price) / COUNT(DISTINCT dc.customer_code), 2) AS
revenue_per_customer
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
GROUP BY dc.market, dc.platform
```

```
ORDER BY dc.market, total_revenue DESC;
```

Insights: The comparative analysis places India in context with other markets and regions. APAC leads in total revenue, with India being a significant contributor within this region. The comparison between India and other markets reveals unique category preferences, helping tailor product assortments. Regional performance analysis shows APAC's dominance, while the top markets by region highlight India's position within APAC. Channel and platform effectiveness analyses show that in India, the Direct channel generates the highest revenue, with Brick & Mortar slightly outperforming E-Commerce. These comparative insights help identify best practices and opportunities for knowledge transfer across markets.

Advanced Analytics

```
-- Calculate customer lifetime value for India
WITH CustomerStats AS (
SELECT
dc.customer_code,
dc.customer,
MIN(fsm.date) AS first_purchase_date,
MAX(fsm.date) AS last_purchase_date,
DATEDIFF(MAX(fsm.date), MIN(fsm.date)) AS customer_lifetime_days,
COUNT(DISTINCT fsm.date) AS purchase_frequency,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_spent,
AVG(fsm.sold_quantity * fgp.gross_price) AS avg_order_value
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dc.customer_code, dc.customer
)
SELECT
customer_code,
customer,
first_purchase_date,
last_purchase_date,
customer_lifetime_days,
purchase_frequency,
total_spent,
avg_order_value,
CASE
   WHEN customer_lifetime_days > 0 THEN (total_spent / customer_lifetime_days) * 365 * 3
   ELSE total_spent * 3
END AS projected_3year_clv
FROM CustomerStats
ORDER BY total_spent DESC
LIMIT 20;

-- Customer churn rate in India
WITH RecentPurchase AS (
SELECT MAX(date) AS most_recent_purchase_date
FROM fact_sales_monthly
),
```

```sql
CutoffDate AS (
SELECT DATE_SUB(most_recent_purchase_date, INTERVAL 1 YEAR) AS cutoff_date
FROM RecentPurchase
),
ChurnedCustomers AS (
SELECT
dc.customer_code,
dc.customer,
MAX(fsm.date) AS last_purchase_date
FROM
dim_customer dc
LEFT JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
WHERE dc.market = 'India'
GROUP BY
dc.customer_code, dc.customer
HAVING
MAX(fsm.date) IS NULL OR MAX(fsm.date) < (SELECT cutoff_date FROM CutoffDate)
)
-- Calculate the churn rate
SELECT
(SELECT COUNT(*) FROM ChurnedCustomers) / (SELECT COUNT(*) FROM dim_customer WHERE market = 'India') *
100 AS churn_rate_india;

-- Identify at-risk customers in India (no purchase in last 6 months)
WITH recent_purchase AS (
SELECT
dc.customer_code
FROM
dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
WHERE
dc.market = 'India' AND
fsm.date >= CURDATE() - INTERVAL 6 MONTH
)
SELECT
dc.customer_code, dc.customer,
MAX(fsm.date) AS last_purchase_date
FROM dim_customer dc
LEFT JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
WHERE dc.market = 'India' AND
dc.customer_code NOT IN (SELECT customer_code FROM recent_purchase)
GROUP BY dc.customer_code, dc.customer
ORDER BY last_purchase_date DESC
LIMIT 20;

-- Product categories frequently purchased together in India
SELECT dp1.category AS category1, dp2.category AS category2, COUNT(*) AS frequency
FROM fact_sales_monthly fsm1
JOIN dim_product dp1 ON fsm1.product_code = dp1.product_code
JOIN fact_sales_monthly fsm2 ON fsm1.customer_code = fsm2.customer_code AND fsm1.date = fsm2.date
JOIN dim_product dp2 ON fsm2.product_code = dp2.product_code
JOIN dim_customer dc ON fsm1.customer_code = dc.customer_code
WHERE fsm1.product_code < fsm2.product_code AND dp1.category <> dp2.category AND dc.market = 'India'
```

```sql
GROUP BY dp1.category, dp2.category
ORDER BY frequency DESC
LIMIT 15;

-- Product divisions frequently purchased together in India
SELECT dp1.division AS division1, dp2.division AS division2, COUNT(*) AS frequency
FROM fact_sales_monthly fsm1
JOIN dim_product dp1 ON fsm1.product_code = dp1.product_code
JOIN fact_sales_monthly fsm2 ON fsm1.customer_code = fsm2.customer_code AND fsm1.date = fsm2.date
JOIN dim_product dp2 ON fsm2.product_code = dp2.product_code
JOIN dim_customer dc ON fsm1.customer_code = dc.customer_code
WHERE fsm1.product_code < fsm2.product_code AND dp1.division <> dp2.division AND dc.market = 'India'
GROUP BY dp1.division, dp2.division
ORDER BY frequency DESC
LIMIT 15;

-- Customer segmentation based on purchasing behavior in India
WITH CustomerStats AS (
SELECT
dc.customer_code,
dc.customer,
COUNT(DISTINCT fsm.date) AS purchase_frequency,
SUM(fsm.sold_quantity * fgp.gross_price) AS total_spent
FROM dim_customer dc
JOIN fact_sales_monthly fsm ON dc.customer_code = fsm.customer_code
JOIN dim_product dp ON fsm.product_code = dp.product_code
JOIN fact_gross_price fgp ON dp.product_code = fgp.product_code
AND fsm.fiscal_year = fgp.fiscal_year
WHERE dc.market = 'India'
GROUP BY dc.customer_code, dc.customer
),
Percentiles AS (
-- compute the 80th and 50th percentile cutoffs using CUME_DIST()
SELECT
MIN(CASE WHEN cd_spent >= 0.8 THEN total_spent END) AS spent_80,
MIN(CASE WHEN cd_spent >= 0.5 THEN total_spent END) AS spent_50,
MIN(CASE WHEN cd_freq >= 0.8 THEN purchase_frequency END) AS freq_80,
MIN(CASE WHEN cd_freq >= 0.5 THEN purchase_frequency END) AS freq_50
FROM (
SELECT
total_spent,
purchase_frequency,
CUME_DIST() OVER (ORDER BY total_spent) AS cd_spent,
CUME_DIST() OVER (ORDER BY purchase_frequency) AS cd_freq
FROM CustomerStats
) x
),
CustomerSegments AS (
SELECT
cs.customer_code,
cs.customer,
cs.purchase_frequency,
cs.total_spent,
CASE
WHEN cs.total_spent >= p.spent_80 AND cs.purchase_frequency >= p.freq_80 THEN 'High Value'
WHEN cs.total_spent >= p.spent_50 AND cs.purchase_frequency >= p.freq_50 THEN 'Medium Value'
ELSE 'Low Value'
END AS customer_segment
FROM CustomerStats cs
CROSS JOIN Percentiles p
)
SELECT
customer_segment,
COUNT(*) AS customer_count,
ROUND(AVG(purchase_frequency), 2) AS avg_purchase_frequency,
ROUND(AVG(total_spent), 2) AS avg_total_spent,
ROUND(SUM(total_spent) / (SELECT SUM(total_spent) FROM CustomerStats) * 100, 2) AS revenue_percentage
FROM CustomerSegments
GROUP BY customer_segment
ORDER BY avg_total_spent DESC;
```

Insights: The advanced analytics provide sophisticated insights into customer behavior and product relationships in India. Customer lifetime value calculations identify the most valuable customers based on historical spending patterns. Churn analysis reveals the percentage of customers who have stopped purchasing and identifies at-risk customers who haven't purchased recently. Product affinity analysis shows which categories and divisions are frequently purchased together, revealing cross-selling opportunities. Customer segmentation divides the customer base into High, Medium, and Low Value groups based on spending and frequency, showing the size and characteristics of each segment. These advanced insights enable highly targeted strategies for customer retention, product bundling, and resource allocation in the Indian market.

Recommendations and Conclusion

Key Findings

India Market Performance:

India represents a significant market for AtliQ Hardware with specific product categories showing strong performance
Customer behavior in India shows distinct patterns in terms of purchase frequency and average order value
All customers are concentrated in the India sub-zone, indicating a focused geographic approach

Product Performance:

Specific product categories consistently perform well across multiple markets
There are clear product affinities that can be leveraged for cross-selling opportunities
Product preferences vary by region, indicating the need for localized strategies

Customer Insights:

Customer segments show distinct purchasing behaviors and lifetime values
Churn analysis reveals customers who have stopped purchasing and those at risk
High-value customers contribute disproportionately to total revenue

Business Recommendations

India-Specific Strategies:

Focus marketing efforts on high-performing product categories in India
Develop targeted promotions for underperforming sub-zones within India
Implement customer retention programs for at-risk customers in the Indian market

Product Portfolio Optimization:

Leverage product affinity insights to create bundled offerings
Consider regional variations when planning new product introductions
Allocate inventory based on regional performance data

Customer Relationship Management:

Implement tiered loyalty programs based on customer segmentation
Develop personalized engagement strategies for high-value customers
Create win-back campaigns for churned customers with high lifetime value potential