End-to-End Data Analytics Project Using MySQL, Power BI, and Excel

---

Submitted by:

Udit Kumar Singh

Tools and Technologies Used:

MySQL (Data Cleaning, Transformation, Reporting Procedures)

Power BI (Dashboard Visualization and Analysis)

Microsoft Excel (Report Validation)

Project Summary

This project demonstrates an end-to-end data analytics workflow, starting with raw data cleaning and transformation in MySQL, moving into optimized data modeling and dashboard creation in Power BI, and concluding with exploratory and validation analyses using Microsoft Excel.

By performing all data cleaning and transformation at the database level, we ensure consistent, scalable, and high-performance reporting across the organization. This approach minimizes redundancy, accelerates Power BI report performance, and provides stakeholders with clean, ready-to-use data without repetitive manual intervention.

The tools integrate seamlessly to deliver a dynamic, efficient, and fully optimized data-driven solution, supporting multiple business domains like Finance, Sales, Marketing, Supply Chain, and Executive KPIs.

Date:

April 2025

Acknowledgments

I would like to express my sincere gratitude to everyone who supported me throughout this project.

Firstly, I would like to thank my **course creator CodeBasics** for providing valuable guidance on creating the SQL procedures and assisting in understanding key concepts for the MySQL database. Their insight helped me approach complex data challenges with a clearer perspective.

I would also like to thank my previous organizations who helped me understand the various aspects of business, who have been an inspiration with their real-world business data use cases, which motivated me to build and optimize the data workflow for meaningful business insights.

Finally, I thank my **friends and family** for their encouragement and support throughout the project. This endeavour wouldn't have been possible without their belief in my abilities.

Part 1:
Data Cleaning

Objective: Ensure consistency, accuracy, and optimal performance by pre-processing the raw data before modeling in Power BI.

- **Remove leading and trailing spaces to ensure data consistency and prevent mismatches during joins and reporting**

  The only column that had trailing spaces was the *customer* column in the *dim_customer* table.

  - UPDATE dim_customer SET customer = TRIM(customer);

- **Correct inconsistencies in data due to case sensitivity and spelling errors to standardize records**
  Some spelling errors were found in some columns of the *dim_market* and *dim_customer* tables.

  - UPDATE dim_customer SET customer = "AtliQ E Store" WHERE customer = "Atliq e Store";
  - UPDATE dim_customer SET customer = "AtliQ Exclusive" WHERE customer = "AltiQ Exclusive";
  - UPDATE dim_customer SET customer = "AtliQ Exclusive" WHERE customer = "Atliq Exclusive";
  - UPDATE dim_market SET sub_zone = "NA" WHERE sub_zone = "nan";
  - UPDATE dim_market SET region = "NA" WHERE region = "nan";

- **Assign the correct sales channel and platform to customers to ensure accurate channel & platform-based reporting**
  Upon exploring the data, it was discovered that the channel for a direct customer (with *customer_code = 90002011*) was mistakenly marked as "Retailer".

  - UPDATE dim_customer SET `channel` = "Direct" WHERE customer_code = 90002011;

- **Handle missing values to improve data quality**
  During data exploration, some values were found to be null, which were then replaced with zero.

  - UPDATE fact_act_est SET forecast_quantity = 0 WHERE forecast_quantity IS NULL;
  - UPDATE fact_act_est SET sold_quantity = 0 WHERE sold_quantity IS NULL;

- **Rename columns for better clarity**
  Some columns were renamed for better understanding and more effective data analysis.

- ALTER TABLE manufacturing_cost RENAME COLUMN cost_year TO fiscal_year;

- **Standardize data types and precision to ensure consistent calculations across datasets**
  For accurate calculations, decimal places were standardized to two decimal places in various columns.
  - ALTER TABLE freight_cost CHANGE COLUMN freight_pct freight_pct DECIMAL(3,2) NOT NULL, CHANGE COLUMN other_cost_pct other_cost_pct DECIMAL(4,3) NOT NULL;
  - ALTER TABLE post_invoice_deductions CHANGE COLUMN discounts_pct discounts_pct DECIMAL(3,2) NOT NULL, CHANGE COLUMN other_deductions_pct other_deductions_pct DECIMAL(3,2) NOT NULL;
  - ALTER TABLE gross_price CHANGE COLUMN gross_price gross_price DECIMAL(5,2) NOT NULL;
  - ALTER TABLE manufacturing_cost CHANGE COLUMN manufacturing_cost manufacturing_cost DECIMAL(5,2) NOT NULL;
  - ALTER TABLE pre_invoice_deductions CHANGE COLUMN pre_invoice_discount_pct pre_invoice_discount_pct DECIMAL(3,2) NOT NULL;

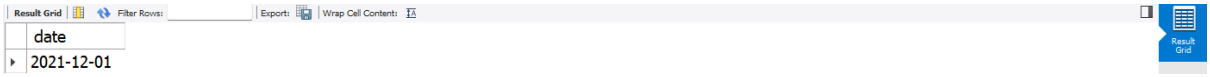
Part 2:
Data Transformation

Objective: Transform and structure the data to optimize it for dynamic reporting and forecasting.

- **Identify the latest available sales date in the dataset to dynamically manage reporting periods and forecasting cut-offs**

  Create a view to fetch the most recent sales date, useful for dynamic filtering and reporting cut-offs

  CREATE VIEW last_sales_month AS SELECT MAX(`date`) AS `date` FROM fact_sales_monthly;



date
2021-12-01

- **Consolidate actual and forecasted sales data into a unified table, adjusting for fiscal years and dynamically calculating gross revenue, net invoice sales, and net sales, to support more efficient and scalable reporting in Power BI**

  Merge historical and forecast data, adjusting for fiscal years, to streamline gross revenue, net sales, and deductions

```
CREATE VIEW fact_actual_est AS
-- creating a cte for appending fact_forecast_monthly with fact_sales_monthly
WITH combined_data AS
(SELECT `date`,YEAR(DATE_ADD(`date`,  INTERVAL 4 MONTH)) AS fiscal_year,
product_code,
customer_code, quantity FROM fact_sales_monthly
UNION ALL
SELECT `date`, YEAR(DATE_ADD(`date`, INTERVAL 4 MONTH)) AS fiscal_year,
product_code,
customer_code, quantity FROM fact_forecast_monthly
WHERE `date` > (SELECT MAX(`date`) from fact_sales_monthly)),
-- creating a cte for calculating gross_revenue
gross_sales AS (SELECT f.`date`, f.fiscal_year, f.product_code, f.customer_code,
f.quantity,
ROUND(g.gross_price,2) AS gross_price,
ROUND((f.quantity * g.gross_price),2) AS gross_revenue
FROM combined_data f
JOIN gdb056.gross_price g
ON g.product_code = f.product_code AND g.fiscal_year = f.fiscal_year),
-- creating a cte for calculating net invoice sales
```

```
net_invoice_sales AS (SELECT gs.`date`, gs.fiscal_year, gs.product_code,
gs.customer_code, gs.quantity,
gs.gross_price, gs.gross_revenue, (gs.gross_revenue*pre.pre_invoice_discount_pct) AS
pre_invoice_discount,
ROUND((gs.gross_revenue * (1 - pre.pre_invoice_discount_pct)),2) AS net_invoice_sales
FROM gross_sales gs
JOIN gdb056.pre_invoice_deductions pre
ON pre.customer_code = gs.customer_code AND pre.fiscal_year = gs.fiscal_year),
-- creating a cte for calculating net sales
net_sales AS(SELECT ns.`date`, ns.fiscal_year, ns.product_code, ns.customer_code,
ns.quantity,
ns.gross_price, ns.gross_revenue, ns.pre_invoice_discount, ns.net_invoice_sales,
(ns.net_invoice_sales*post.discounts_pct) AS post_invoice_deduction,
(ns.net_invoice_sales*post.other_deductions_pct) AS other_deductions
FROM net_invoice_sales ns
JOIN gdb056.post_invoice_deductions post
ON post.`date`=ns.`date` AND post.customer_code=ns.customer_code AND
post.product_code=ns.product_code)
-- SELECT statement for cte scope that will give us the final result,
SELECT `date`, fiscal_year, product_code, customer_code, quantity, gross_price,
gross_revenue, pre_invoice_discount,
net_invoice_sales, post_invoice_deduction, other_deductions FROM net_sales;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

date	fiscal_year	customer_code	customer	market	product_code	product	variant	sold_quantity	gross
2017-09-01	2018	70002017	Atliq Exclusive	India	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	51	15.40
2017-09-01	2018	70002018	Atliq e Store	India	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	77	15.40
2017-09-01	2018	70003181	Atliq Exclusive	Indonesia	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	17	15.40
2017-09-01	2018	70003182	Atliq e Store	Indonesia	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	6	15.40
2017-09-01	2018	70006157	Atliq Exclusive	Philiphines	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	5	15.40
2017-09-01	2018	70006158	Atliq e Store	Philiphines	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	7	15.40
2017-09-01	2018	70007198	Atliq Exclusive	South Korea	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	29	15.40
2017-09-01	2018	70007199	Atliq e Store	South Korea	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	34	15.40
2017-09-01	2018	70008169	Atliq Exclusive	Australia	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	22	15.40
2017-09-01	2018	70008170	Atliq e Store	Australia	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	5	15.40
2017-09-01	2018	70011193	Atliq Exclusive	France	A0118150101	AQ Dracula HDD – 3.5 Inc...	Standard	10	15.40

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

- **Create a dynamic Date Dimension table that accounts for fiscal calendar adjustments (September-August) and extends the calendar for future forecast periods, enabling accurate time-based analysis**

  Build a date table that adjusts fiscal months starting from September, used for filtering and aggregations

```
CREATE VIEW dim_date AS SELECT DISTINCT `date`,YEAR(DATE_ADD(`date`,
INTERVAL 4 MONTH)) AS fiscal_year,
DATE_FORMAT(date, '%Y-%m-01') AS start_of_month, MONTH(DATE_ADD(`date`,
INTERVAL 4 MONTH)) AS fiscal_month,
CONCAT("Q",CEILING(MONTH(DATE_ADD(`date`, INTERVAL 4 MONTH))/3)) AS
`quarter` FROM fact_sales_monthly
UNION ALL  -- we'll append another table below
```

```
SELECT DISTINCT `date`, YEAR(DATE_ADD(`date`, INTERVAL 4 MONTH)) AS
fiscal_year,
DATE_FORMAT(date, '%Y-%m-01') AS start_of_month, MONTH(DATE_ADD(`date`,
INTERVAL 4 MONTH)) AS fiscal_month,
CONCAT("Q",CEILING(MONTH(DATE_ADD(`date`, INTERVAL 4 MONTH))/3)) AS
`quarter` FROM fact_forecast_monthly
WHERE `date` > (SELECT MAX(`date`) from fact_sales_monthly);
```

date	fiscal_year	start_of_month	fiscal_month	quarter
2017-09-01	2018	2017-09-01	1	Q1
2017-10-01	2018	2017-10-01	2	Q1
2017-11-01	2018	2017-11-01	3	Q1
2017-12-01	2018	2017-12-01	4	Q2
2018-01-01	2018	2018-01-01	5	Q2
2018-02-01	2018	2018-02-01	6	Q2
2018-03-01	2018	2018-03-01	7	Q3
2018-04-01	2018	2018-04-01	8	Q3
2018-05-01	2018	2018-05-01	9	Q3
2018-06-01	2018	2018-06-01	10	Q4
2018-07-01	2018	2018-07-01	11	Q4
2018-08-01	2018	2018-08-01	12	Q4

- **Generate a fiscal year reference table that clearly labels the current or upcoming fiscal year as "EST" (Estimated), to distinguish actual vs. forecast periods in dashboard visuals**

  Label the latest fiscal year as "EST" to distinguish between actuals and estimates

```
CREATE VIEW fiscal_year AS
SELECT fiscal_year,
CASE
WHEN fiscal_year = (SELECT MAX(fiscal_year) FROM fact_actual_est) THEN
CONCAT(fiscal_year, ' EST')
ELSE fiscal_year
END AS fy_desc
FROM (SELECT DISTINCT fiscal_year FROM fact_actual_est) AS fy_list;
```

fiscal_year	fy_desc
2018	2018
2019	2019
2020	2020
2021	2021
2022	2022 EST

- **Combine actual sold quantities and forecasted quantities into a single fact table to enable side-by-side analysis of past performance and future expectations**

  Merge actual and forecast quantities for comparative analysis

  ```
  CREATE VIEW fact_quantity AS
  SELECT s.`date`, YEAR(DATE_ADD(s.`date`, INTERVAL 4 MONTH)) AS fiscal_year, ...
  FROM fact_sales_monthly s
  LEFT JOIN fact_forecast_monthly f
  ON s.`date` = f.`date` AND s.product_code = f.product_code AND s.customer_code =
  f.customer_code;
  ```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

date	fiscal_year	product_code	customer_code	sold_quantity	forecast_quantity
2017-09-01	2018	A6218160101	70008169	81	146
2017-09-01	2018	A6218160101	90008165	157	120
2017-09-01	2018	A6218160101	90008166	126	216
2017-09-01	2018	A6218160101	90008167	160	141
2017-09-01	2018	A6218160101	70008170	120	85
2017-09-01	2018	A6218160101	90027207	9	14
2017-09-01	2018	A6218160101	70023031	9	30
2017-09-01	2018	A6218160101	90023022	24	8
2017-09-01	2018	A6218160101	90023025	22	25
2017-09-01	2018	A6218160101	90023026	37	17
2017-09-01	2018	A6218160101	90023027	4	39
2017-09-01	2018	A6218160101	90023029	12	25

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

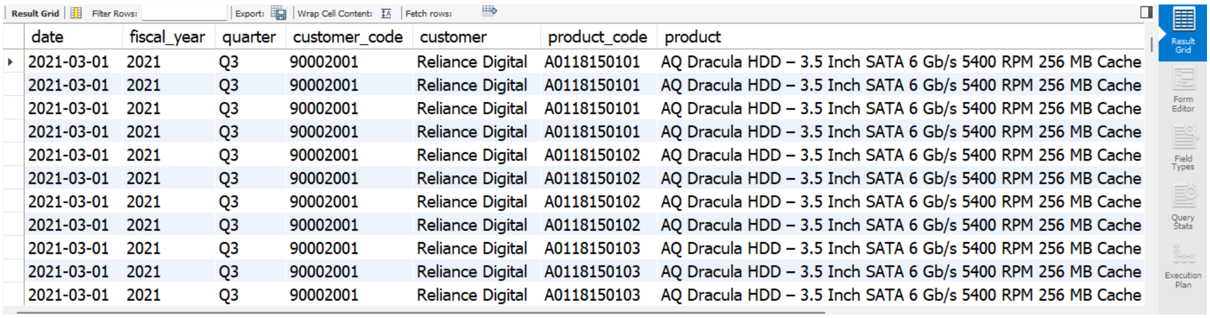
Part 3:
Procedure Reports

Objective: Create SQL Stored Procedures to automate reporting processes and minimize manual query execution.

- **Retrieve monthly sales data filtered by fiscal year, selected customers, and quarter**

The procedure returns detailed sales figures, including sold quantity, gross price, and total sales per product

```
DELIMITER $$
CREATE PROCEDURE get_monthly_sales_new(IN `fy` INT, IN `cust_code`
VARCHAR(255), IN `qtr` VARCHAR(5))
BEGIN
            SELECT * FROM
   (SELECT s.`date`,
   YEAR(DATE_ADD(`date`, INTERVAL 4 MONTH)) AS fiscal_year,
   CONCAT("Q",CEILING(MONTH(DATE_ADD(`date`,INTERVAL 4 MONTH))/3)) AS
`quarter`,
   c.customer_code, c.customer, p.product_code, p.product, p.variant,
s.sold_quantity,
   ROUND(g.gross_price,2) AS gross_price,
   ROUND((s.sold_quantity*g.gross_price),2) AS total_price
            FROM (((fact_sales_monthly s JOIN dim_customer c ON
s.customer_code=c.customer_code)
   JOIN dim_product p ON s.product_code=p.product_code)
   JOIN fact_gross_price g ON p.product_code=g.product_code)) AS croma_data
   WHERE fiscal_year = `fy` AND FIND_IN_SET(customer_code,`cust_code`) >0 AND
`quarter` = `qtr` ORDER BY `date` ASC;
END$$
```



date	fiscal_year	quarter	customer_code	customer	product_code	product
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150101	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150102	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150102	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150102	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150102	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150103	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150103	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache
2021-03-01	2021	Q3	90002001	Reliance Digital	A0118150103	AQ Dracula HDD – 3.5 Inch SATA 6 Gb/s 5400 RPM 256 MB Cache

- **Identify the top-performing markets based on net sales for a selected fiscal year**

  The procedure dynamically ranks markets and returns the top N markets as specified

```
DELIMITER $$
CREATE PROCEDURE top_markets(IN `top` INT, IN `fy` INT)
BEGIN
        SELECT fiscal_year, market, net_sales FROM (SELECT market, fiscal_year,
         ROUND(SUM(`net_sales`)/1000000,2) AS net_sales,
         DENSE_RANK() OVER(PARTITION BY fiscal_year ORDER BY SUM(`net_sales`)
DESC)
        AS row_num FROM get_net_sales GROUP BY market, fiscal_year) AS new_table
                WHERE row_num <= `top` AND fiscal_year = `fy`;
END$$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fiscal_year	market	net_sales
2021	India	210.50
2021	USA	131.87
2021	South Korea	64.16

- **Retrieve the top customers ranked by net sales for a selected fiscal year**

  Enables performance evaluation of the highest-contributing customers within a specific year

```
DELIMITER $$
CREATE PROCEDURE top_customers(IN top INT, IN fy INT)
BEGIN
                SELECT fiscal_year, customer_code, customer, net_sales FROM (SELECT
fiscal_year, customer_code,
    customer, ROUND(SUM(`net_sales`)/1000000,2) AS net_sales,
                DENSE_RANK() OVER(PARTITION BY fiscal_year ORDER BY
SUM(`net_sales`) DESC)
    AS row_num FROM get_net_sales GROUP BY customer_code, customer, fiscal_year)
AS new_table
                WHERE row_num <= top AND fiscal_year = fy;
END$$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fiscal_year	customer_code	customer	net_sales
2021	80007195	Sage	22.91
2021	80007196	Leader	22.44
2021	80001019	Neptune	20.10

- **Identify top-selling products based on net sales, customer-wise, for a selected fiscal year**

  Helps in product performance analysis at the customer level

```
DELIMITER $$
CREATE PROCEDURE top_products(IN top INT, IN fy INT)
BEGIN
                SELECT fiscal_year, product_code, product, net_sales FROM (SELECT fiscal_year, customer_code, product_code, product,
    ROUND(SUM(`net_sales`)/1000000,2) AS net_sales,
                DENSE_RANK() OVER(PARTITION BY fiscal_year, customer_code ORDER BY SUM(`net_sales`) DESC)
    AS row_num FROM get_net_sales GROUP BY product_code, product, fiscal_year, customer_code) AS new_table
                WHERE row_num <= top AND fiscal_year = fy;
END$$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fiscal_year	product_code	product	net_sales
2021	A1521150601	AQ Electron 3 3600 Desktop Processor	0.16
2021	A1420150501	AQ Electron 4 3600 Desktop Processor	0.14
2021	A1421150503	AQ Electron 4 3600 Desktop Processor	0.12
2021	A1521150602	AQ Electron 3 3600 Desktop Processor	0.12
2021	A1521150601	AQ Electron 3 3600 Desktop Processor	0.10
2021	A1320150402	AQ Electron 5 3600 Desktop Processor	0.10
2021	A1521150601	AQ Electron 3 3600 Desktop Processor	0.07
2021	A1421150503	AQ Electron 4 3600 Desktop Processor	0.07
2021	A1521150602	AQ Electron 3 3600 Desktop Processor	0.06
2021	A1521150601	AQ Electron 3 3600 Desktop Processor	0.05
2021	A7321160302	AQ Wi Power Dx3	0.05
2021	A1420150501	AQ Electron 4 3600 Desktop Processor	0.05

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

- **Build a hierarchical analysis by selecting the top markets, their corresponding top customers, and their corresponding top products based on net sales in a selected fiscal year**

  This step-by-step temporary table creation and final aggregation allow detailed and targeted sales performance reporting

```
DELIMITER $$
CREATE PROCEDURE topentitiesdetails(IN market_n INT, IN customer_n INT, IN product_n INT, IN fy INT)
BEGIN
  -- Step 1: Get top markets
  CREATE TEMPORARY TABLE temp_top_markets AS
  SELECT market FROM (SELECT market,
                DENSE_RANK() OVER (ORDER BY SUM(net_sales) DESC) AS rk
    FROM get_net_sales WHERE fiscal_year = fy
    GROUP BY market) ranked_markets
    WHERE rk <= market_n;

  -- Step 2: Get top customers within top markets
```

```
  CREATE TEMPORARY TABLE temp_top_customers AS
  SELECT customer_code, market FROM (SELECT customer_code,
                  market, DENSE_RANK() OVER (PARTITION BY market ORDER BY
SUM(net_sales) DESC) AS rk
    FROM get_net_sales WHERE fiscal_year = fy
                  AND market IN (SELECT market FROM temp_top_markets)
    GROUP BY customer_code, market) ranked_customers
                  WHERE rk <= customer_n;

  -- Step 3: Get top products within top customers
  CREATE TEMPORARY TABLE temp_top_products AS
  SELECT product_code, customer_code, market FROM (SELECT product_code,
                  customer_code, market,
                  DENSE_RANK() OVER (PARTITION BY customer_code ORDER BY
SUM(net_sales) DESC) AS rk
    FROM get_net_sales WHERE fiscal_year = fy
                  AND (customer_code, market) IN (SELECT customer_code, market
FROM temp_top_customers)
    GROUP BY product_code, customer_code, market) ranked_products
                  WHERE rk <= product_n;

  -- Step 4: Final result
  SELECT
  g.fiscal_year, g.market, g.customer_code, customer, g.product_code, product,
  SUM(g.sold_quantity) AS quantity,
  ROUND(AVG(g.gross_price), 2) AS gross_price,
  ROUND(SUM(g.total_amount)/1000000, 2) AS gross_revenue,
  ROUND(AVG(g.pre_invoice_discount_pct)*100, 2) AS pre_invoice_deductions_pct,
  ROUND(SUM(g.net_invoice_sales)/1000000, 2) AS net_invoice_sales,
  ROUND(AVG(g.discounts_pct)*100, 2) AS post_invoice_deductions_pct,
  ROUND(AVG(g.other_deductions_pct)*100, 2) AS other_deductions_pct,
  ROUND(SUM(g.net_sales)/1000000, 2) AS net_sales
  FROM get_net_sales g
  JOIN temp_top_products p ON g.market = p.market
        AND g.customer_code = p.customer_code AND g.product_code =
p.product_code
  WHERE g.fiscal_year = fy GROUP BY g.fiscal_year, g.market, g.customer_code,
g.product_code;

  -- Cleanup
  DROP TEMPORARY TABLE temp_top_markets;
  DROP TEMPORARY TABLE temp_top_customers;
  DROP TEMPORARY TABLE temp_top_products;
END $$
```

fiscal_year	market	customer_code	customer	product_code	product	quantity	gr
2021	India	70002017	Atliq Exclusive	A1521150601	AQ Electron 3 3600 Desktop Processor	1489	18
2021	India	70002017	Atliq Exclusive	A1420150501	AQ Electron 4 3600 Desktop Processor	1536	16
2021	India	90002008	Amazon	A1521150601	AQ Electron 3 3600 Desktop Processor	2208	18
2021	India	90002008	Amazon	A1521150602	AQ Electron 3 3600 Desktop Processor	2171	18
2021	USA	90022074	Flipkart	A1320150403	AQ Electron 5 3600 Desktop Processor	1363	16
2021	USA	90022074	Flipkart	A1421150503	AQ Electron 4 3600 Desktop Processor	1270	17
2021	USA	90022081	Amazon	A1521150602	AQ Electron 3 3600 Desktop Processor	1425	18
2021	USA	90022081	Amazon	A1521150601	AQ Electron 3 3600 Desktop Processor	1375	18

- **Retrieve top customers within a specific market based on net sales for a selected fiscal year**

  Enables market-specific customer performance reporting

```
DELIMITER $$
CREATE PROCEDURE top_customers_per_market(IN top INT, IN fy INT, IN input_market
VARCHAR(50))
BEGIN
                SELECT fiscal_year, market, customer_code, customer,
ROUND(net_sales/1000000, 2) AS net_sales
    FROM (SELECT fiscal_year, customer_code, customer, market,
     SUM(net_sales) AS net_sales,
                DENSE_RANK() OVER(PARTITION BY fiscal_year, market ORDER BY
SUM(net_sales) DESC)
    AS row_num FROM get_net_sales
    GROUP BY customer_code, customer, fiscal_year, market) AS new_table
                WHERE row_num <= top AND fiscal_year = fy AND market =
input_market;
END$$
```

fiscal_year	market	customer_code	customer	net_sales
2021	USA	90022081	Amazon	11.90
2021	USA	90022074	Flipkart	10.36
2021	USA	70022084	Atliq Exclusive	10.21

- Retrieve top products sold to a specific customer based on net sales for a selected fiscal year

  Allows a customer-centric product performance drill-down

```
DELIMITER $$
CREATE PROCEDURE top_products_per_market(IN top INT, IN fy INT, IN input_customer VARCHAR(50))
BEGIN
                SELECT fiscal_year, customer_code, customer, product_code, product,
    ROUND(net_sales/1000000, 2) AS net_sales
    FROM (SELECT fiscal_year, customer_code, customer, product_code, product,
     SUM(net_sales) AS net_sales,
                DENSE_RANK() OVER(PARTITION BY fiscal_year, customer ORDER BY SUM(net_sales) DESC)
    AS row_num FROM get_net_sales
    GROUP BY customer_code, customer, fiscal_year, product_code, product) AS new_table
                WHERE row_num <= top AND fiscal_year = fy AND customer = input_customer;
END$$
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

fiscal_year	customer_code	customer	product_code	product	net_sales
2021	90002002	Croma	A1521150601	AQ Electron 3 3600 Desktop Processor	0.09
2021	90002002	Croma	A1319150401	AQ Electron 5 3600 Desktop Processor	0.09
2021	90002002	Croma	A1219150302	AQ 5000 Series Ultron 8 5900X Desktop Processor	0.08

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan