DATABASE MANAGEMENT

Project

Project By
Sergiu Hotea

https://www.linkedin.com/in/sergiuhotea
https://github.com/shotea

Project Overview

Problem Statement

TechGear Inc. is navigating the complex task of refining its supply chain operations to reduce cost and mitigate risks to uphold a competitive edge in the fast-paced high-tech gadget market.

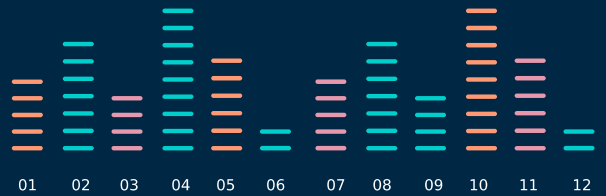
To achieve this TechGear Inc. must address the following crucial challenges:

Challenges

Quality Control and Cost Reduction
Cost Optimization in Manufacturing
Supply Chain Optimization

Data Description

In a nutshell, our database setup allows us to dive deep into our manufacturing process.
We can identify quality issues, analyze manufacturing costs, and gain insights into the assembly process—all crucial elements in ensuring a smooth and efficient supply chain.



Attributes

TechGear Inc.'s business essence, goals, and challenges per the problem statement

table_name	col_name	type	atomic	repeating_group	pk	fk	anomalies
Components Table	component_Id	VARCHAR(255)	YES	NO	YES	NO	NO
	component	VARCHAR(255)	YES	NO	NO	NO	NO
	Products	VARCHAR(255)	YES	NO	NO	NO	NO
Manufacturers Table	manufacturer_ID	VARCHAR(255)	YES	NO	YES	NO	NO
	manufacturer	VARCHAR(255)	YES	NO	NO	NO	NO
Quality_Control Table	component_Id	VARCHAR(255)	YES	NO	YES	YES	NO
	Quality_ontrol_type	VARCHAR(255)	YES	NO	YES	NO	NO
	result: pass_or_fail	VARCHAR(255)	YES	NO	NO	NO	NO
	test_number	VARCHAR(255)	YES	NO	YES	NO	NO
Manufacturer_cost Table	manufacturer_id	VARCHAR(255)	YES	NO	YES	NO	NO
	manufacturing_cost	DECIMAL(10, 2)	YES	NO	NO	NO	NO
Quantity_Details Table	manufacturer_id	VARCHAR(255)	YES	NO	YES	YES	NO
	quanty	VARCHAR(255)	YES	NO	NO	NO	NO
Assembling_Plant Table	assembling_plant_id	VARCHAR(255)	YES	NO	YES	NO	NO
	assembly_cost	VARCHAR(255)	YES	NO	NO	NO	NO
	product	DECIMAL(10, 2)	YES	NO	NO	NO	NO
	assembly_plant	DECIMAL(10, 2)	YES	NO	NO	NO	NO

Entity-Relationship Diagram (ERD)

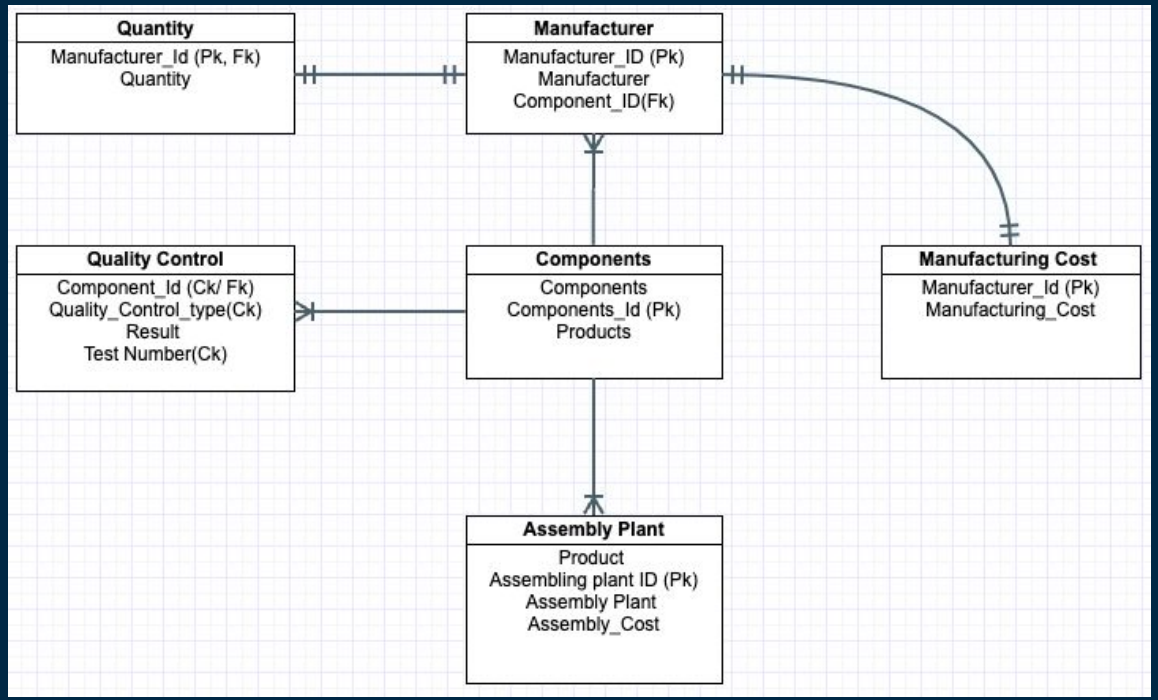


Diagram Overview:
The ERD visually represents these entities as boxes and their relationships as lines between the boxes. Each entity has its attributes listed within the box, and the lines show how the entities are connected based on their relationships.

This ERD provides a clear visualization of the database structure, displaying how different entities are related to each other through their keys and relationships.

RAW DATA LOADING

01

```
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS spm;
Query OK, 0 rows affected, 1 warning (0.001 sec)

MariaDB [(none)]> use spm;
Database changed
MariaDB [spm]> CREATE TABLE dataraw (
    ->     Product VARCHAR(255),
    ->     Component_Id INT,
    ->     Component VARCHAR(255),
    ->     Quality_Control_Type VARCHAR(255),
    ->     Result VARCHAR(255),
    ->     Reason VARCHAR(255),
    ->     Manufacturer_ID VARCHAR(255),
    ->     Manufacturer VARCHAR(255),
    ->     Manufacturing_Cost DECIMAL(10, 2),
    ->     Assembly_Plant VARCHAR(255),
    ->     Assembly_Cost DECIMAL(10, 2),
    ->     Assembling_Product VARCHAR(255),
    ->     Assembling_Plant_ID VARCHAR(255)
    -> );
```

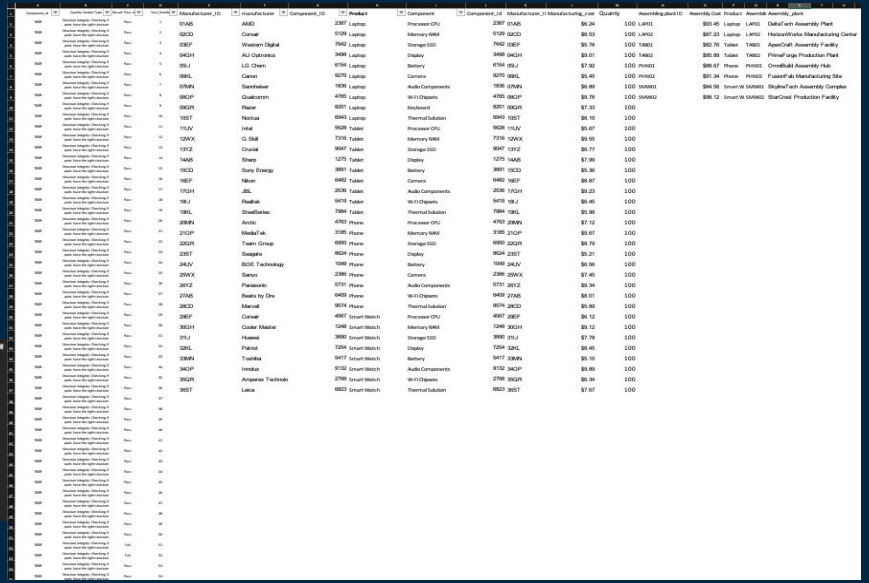
```
LOAD DATA LOCAL INFILE 'C:/Program Files/MariaDB 11.3/data/dataraw.csv'
    INTO TABLE dataraw
    FIELDS TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '"'
    LINES TERMINATED BY '\r\n'
    IGNORE 1 ROWS;
```

02

Creating a MariaDB table named dataraw for electronic components in the 'spm' database.

Data is loaded from a CSV file 'dataraw.csv' with standard formatting. This sets up a database to manage and analyze electronic component details effectively.

The dataset as we can see contains multiple fields that can be organized into separate tables based on their unique characteristics. Here's an attempt to normalize the data...



1NF conformant

01

```
-- Create the table for Quality and Manufacturing data
CREATE TABLE quality_manufacturing (
    Product VARCHAR(255),
    Component_Id INT,
    Component VARCHAR(255),
    Quality_Control_Type VARCHAR(255),
    Result VARCHAR(255),
    Reason VARCHAR(255),
    Manufacturer_ID VARCHAR(255),
    Manufacturer VARCHAR(255),
    Manufacturing_Cost DECIMAL(10, 2)
);
```

To transform the data into the first normal form (1NF), we need to ensure that each attribute contains atomic (indivisible) values, and there are no repeating groups within a row.

Tables in 1NF:

Components Table:
- Component_ID (Primary Key)
- Manufacturer_ID (Foreign Key)

Quality_Control Table:
- Test_Number (Primary Key)
- Component_ID (Foreign Key)

Manufacturing_Cost Table:
- Manufacturer_ID (Primary Key)

Quantity Table:
- Manufacturer_ID (Primary Key)

Assembling_Plant Table:
- Assembling_Plant_ID (Primary Key)

```
-- Insert data into the Quality and Manufacturing table
INSERT INTO quality_manufacturing (Product, Component_Id, Component, Quality_Control_Type, Result, Reason, Manufacturer_ID, Manufacturer,
Manufacturing_Cost)
SELECT Product, Component_Id, Component, Quality_Control_Type, Result, Reason, Manufacturer_ID, Manufacturer, Manufacturing_Cost
FROM dataraw;

-- Retrieve and view data from the Quality and Manufacturing table
SELECT * FROM quality_manufacturing;
```

02

```
-- Create the table for Assembly data
CREATE TABLE assembly (
    Assembly_Plant VARCHAR(255),
    Assembly_Cost DECIMAL(10, 2),
    Assembling_Product VARCHAR(255),
    Assembling_Plant_ID VARCHAR(255)
);
```

As a result each table now contains atomic values, and there are no repeating groups within each row. Relationships between tables are established through foreign key-primary key relationships.

```
-- Insert data into the Assembly table
INSERT INTO assembly (Assembly_Plant, Assembly_Cost, Assembling_Product, Assembling_Plant_ID)
SELECT Assembly_Plant, Assembly_Cost, Assembling_Product, Assembling_Plant_ID
FROM dataraw;
```

2NF & 3NF conformant

```
-- Create Quality Control table
CREATE TABLE IF NOT EXISTS quality_control (
    component_id INT,
    quality_control_type VARCHAR(255),
    result VARCHAR(255),
    reason VARCHAR(255)
);

-- Insert data into Quality Control table
INSERT IGNORE INTO quality_control (component_id, quality_control_type, result, reason)
SELECT Component_Id, Quality_Control_Type, Result, Reason
FROM quality_manufacturing;

-- Create Manufacturer table
CREATE TABLE IF NOT EXISTS manufacturer (
    manufacturer_id VARCHAR(255),
    manufacturer VARCHAR(255)
);

-- Insert data into Manufacturer table
INSERT IGNORE INTO manufacturer (manufacturer_id, manufacturer)
SELECT Manufacturer_ID, Manufacturer
FROM quality_manufacturing;

-- Create Components table
CREATE TABLE IF NOT EXISTS components (
    component_id INT,
    product VARCHAR(255),
    component VARCHAR(255)
);

-- Insert data into Components table
INSERT IGNORE INTO components (component_id, product, component)
SELECT Component_Id, Product, Component
FROM quality_manufacturing;

-- Create Manufacturing Cost table
CREATE TABLE IF NOT EXISTS manufacturing_cost (
    manufacturer_id VARCHAR(255),
    manufacturing_cost DECIMAL(10, 2)
);

-- Insert data into Manufacturing Cost table
INSERT IGNORE INTO manufacturing_cost (manufacturer_id, manufacturing_cost)
SELECT Manufacturer_ID, Manufacturing_Cost
FROM quality_manufacturing;
```

To achieve 2NF, we needed to ensure that all attributes are fully dependent on the entire primary key.

Transformed Tables in 2NF:

Components Table (2NF): Component_ID (PK)
Manufacturers Table (2NF): Manufacturer_ID (PK)
Quality_Control Table (2NF): Test_Number (PK), Component_ID (FK)
Manufacturing_Cost Table (2NF): Manufacturer_ID (PK)
Quantity_control Table (2NF): Manufacturer_ID (PK)
Assembling_Plant Table (2NF): Assembling_Plant_ID (PK)

To reach the third normal form (3NF) from the 2NF tables, we made sure that there was no transitive dependencies (Meaning that a non-primary attributes rely on other non-primary attributes within the same table).

```
-- Create Assembling Cost table
CREATE TABLE IF NOT EXISTS assembling_cost (
    assembling_plant_id VARCHAR(255),
    assembly_cost DECIMAL(10, 2)
);

-- Insert data into Assembling Cost table
INSERT IGNORE INTO assembling_cost (assembling_plant_id, assembly_cost)
SELECT Assembling_Plant_ID, Assembly_Cost
FROM dataraw;

-- Create Assembly Plant table
CREATE TABLE IF NOT EXISTS assembly_plant (
    assembling_product VARCHAR(255),
    assembling_plant_id VARCHAR(255),
    assembly_plant VARCHAR(255)
);

-- Insert data into Assembly Plant table
INSERT IGNORE INTO assembly_plant (assembling_product, assembling_plant_id, assembly_plant)
SELECT Assembling_Product, Assembling_Plant_ID, Assembly_Plant
FROM dataraw;
```

Query Tests

01 Manufacturers facing quality issues, including failure reasons and components affected.

```
MariaDB [spm]> SELECT
    ->     qm.manufacturer_id,
    ->     qm.manufacturer,
    ->     qc.component_id,
    ->     qc.quality_control_type,
    ->     qc.result,
    ->     qc.reason
    -> FROM
    ->     quality_control qc
    -> JOIN
    ->     quality_manufacturing qm ON qc.component_id = qm.component_id
    -> WHERE
    ->     qc.result = 'Fail';
```

manufacturer_id	manufacturer	component_id	quality_control_type	result	reason
03EF	Storage SSD: Western Digital	7642	Structure Integrity: Ch	Fail	SSD fails to read or write data, or its speed is significantly below expectations
05IJ	Battery: LG Chem	6154	Structure Integrity: Ch	Fail	Battery capacity significantly degrades, leading to short usage times or failure to hold a charge.
06KL	Camera: Canon	9270	Structure Integrity: Ch	Fail	The camera produces blurry, distorted, or low-quality images or fails to function.
08OP	Wi-Fi Chipsets: Qualcomm	4765	Structure Integrity: Ch	Fail	Wi-Fi chipset fails to connect, drops connections, or delivers slow speeds.
11UV	Processor CPU: Intel	5628	Structure Integrity: Ch	Fail	The laptop's CPU underperforms, overheats, or crashes during benchmark tests.
14AB	Display: Sharp	1275	Structure Integrity: Ch	Fail	The laptop's display has dead pixels, flickers, or displays distorted images.
15CD	Battery: Sony Energy	3891	Structure Integrity: Ch	Fail	The laptop's battery capacity significantly degrades, leading to short usage times or failure to hold a charge

The query displays the average manufacturing cost for each manufacturer, helping identify manufacturers with lower average costs for potential cost optimization in manufacturing.

02

```
MariaDB [spm]> SELECT manufacturer_id, manufacturer, AVG(manufacturing_cost) AS avg_cost
    -> FROM quality_manufacturing
    -> GROUP BY manufacturer_id, manufacturer
    -> ORDER BY avg_cost ASC;
```

manufacturer_id	manufacturer	avg_cost
33MN	Storage SSD: Toshiba	5.100000
23ST	Storage SSD: Seagate	5.210000
15CD	Battery: Sony Energy	5.360000
06KL	Camera: Canon	5.450000
11UV	Processor CPU: Intel	5.670000
03EF	Storage SSD: Western Digital	5.780000
28CD	Wi-Fi Chipsets: Marvell	5.890000
19KL	Keyboard: SteelSeries	5.980000
29EF	Keyboard: Corsair (again)	6.120000

03 Retrieve Quality and Manufacturing Data:

```
MariaDB [spm]> SELECT * FROM quality_manufacturing;\
```

product	component_id	component	quality_control_type	result	reason	manufacturer_id	manufacturer	manufacturing_cost
Laptop	2387	Processor CPU	Structure Integrity: Checking if parts have the right structure	Pass	The CPU does not cause system crashes or errors during testing.	01AB	Processor CPU: AMD	0.00
Laptop	5129	Memory RAM	Structure Integrity: Checking if parts have the right structure	Pass	RAM successfully completes a memory test without errors.	02CD	Memory RAM: Corsair	0.00
Laptop	7642	Storage SSD	Structure Integrity: Checking if parts have the right structure	Fail	SSD fails to read or write data, or its speed is significantly below expectations	03EF	Storage SSD: Western Digital	0.00
Laptop	3498	Display	Structure Integrity: Checking if parts have the right structure	Pass	Display exhibits vibrant colors, sharp resolution, and no dead pixels.	04GH	Display: AU Optronics	0.00
Laptop	6154	Battery	Structure Integrity: Checking if parts have the right structure	Fail	Battery capacity significantly degrades, leading to short usage times or failure to hold a charge.	05IJ	Battery: LG Chem	0.00
Laptop	9270	Camera	Structure Integrity: Checking if parts have the right structure	Fail	The camera produces blurry, distorted, or low-quality images or fails to function.	06KL	Camera: Canon	0.00
Laptop	1836	Audio Components	Structure Integrity: Checking if parts have the right structure	Pass	Audio components (speakers or headphones) provide clear, distortion-free sound.	07MN	Audio Components: Sennheiser	0.00
Laptop	4765	Wi-Fi Chipsets	Structure Integrity: Checking if parts have the right structure	Fail	Wi-Fi chipset fails to connect, drops connections, or delivers slow speeds.	08OP	Wi-Fi Chipsets: Qualcomm	0.00
Laptop	8201	Keyboard	Structure Integrity: Checking if parts have the right structure	Pass	The keyboard registers all keypresses accurately and feels comfortable to use.	09QR	Keyboard: Razer	0.00
Laptop	6943	Thermal Solution	Structure Integrity: Checking if parts have the right structure	Pass	The thermal solution (cooling system) keeps the component temperatures within acceptable limits, preventing overheating.	10ST	Thermal Solution: Noctua	0.00
Tablet	5628	Processor CPU	Structure Integrity: Checking if parts have the right structure	Fail	The laptop's CPU underperforms, overheats, or crashes during benchmark tests.	11UV	Processor CPU: Intel	0.00
Tablet	7316	Memory RAM	Structure Integrity: Checking if parts have the right structure	Pass	The laptop's RAM passes a memory test without errors.	12WX	Memory RAM: G.Skill	0.00
Tablet	9047	Storage SSD	Structure Integrity: Checking if parts have the right structure	Pass	The laptop's SSD passes read and write speed benchmark tests with expected performance.	13YZ	Storage SSD: Crucial	0.00
Tablet	1275	Display	Structure Integrity: Checking if parts have the right structure	Fail	The laptop's display has dead pixels, flickers, or displays distorted images.	14AB	Display: Sharp	0.00
Tablet	3891	Battery	Structure Integrity: Checking if parts have the right structure	Fail	The laptop's battery capacity significantly degrades, leading to short usage times or failure to hold a charge	15CD	Battery: Sony Energy	0.00
Tablet	6482	Camera	Structure Integrity: Checking if parts have the right structure	Pass	The laptop's built-in camera captures clear and sharp images and video without distortions.	16EF	Camera: Nikon	0.00
Tablet	2536	Audio Components	Structure Integrity: Checking if parts have the right structure	Pass	The laptop's audio components (speakers or headphones) provide clear, distortion-free sound.	17GH	Audio Components: JBL	0.00
Tablet	5419	Wi-Fi Chipsets	Structure Integrity: Checking if parts have the right structure	Pass	The laptop's Wi-Fi chipset connects to networks, maintains a stable connection, and provides expected speeds.	18IJ	Wi-Fi Chipsets: Realtek	0.00
Tablet	7984	Thermal Solution	Structure Integrity: Checking if parts have the right structure	Pass	The laptop's thermal solution (cooling system) keeps the component temperatures within acceptable limits, preventing overheating.	19KL	Keyboard: SteelSeries	0.00
Phone	4763	Processor CPU	Structure Integrity: Checking if parts have the right structure	Pass	The CPU is structurally sound, functioning efficiently without structural problems.	20MN	Thermal Solution: Arctic	0.00
Phone	3185	Memory RAM	Structure Integrity: Checking if parts have the right structure	Pass	The RAM modules have a sound structure, functioning correctly without structural problems.	21OP	Processor CPU: MediaTek	0.00

Query Tests

04 QC&CR: Identifying failed quality control components with associated details, aiding cost reduction efforts.

```
MariaDB [spm]> SELECT
    ->     qm.component_id,
    ->     qm.component,
    ->     qc.quality_control_type,
    ->     qc.result,
    ->     qc.reason,
    ->     qm.manufacturer,
    ->     qm.manufacturer_id,
    ->     qm.manufacturing_cost
    -> FROM
    ->     quality_control qc
    -> JOIN
    ->     quality_manufacturing qm ON qc.component_id = qm.component_id
    -> WHERE
    ->     qc.result = 'Fail';
```

component_id	component	quality_control_type	result	reason	manufacturer	manufacturer_id	manufacturing_cost
7642	Storage SSD	Structure Integrity: Checking if parts have the right structure	Fail	SSD fails to read or write data, or its speed is significantly below expectations	Storage SSD: Western Digital	03EF	5.78
6154	Battery	Structure Integrity: Checking if parts have the right structure	Fail	Battery capacity significantly degrades, leading to short usage times or failure to hold a charge.	Battery: LG Chem	05IJ	7.92
9270	Camera	Structure Integrity: Checking if parts have the right structure	Fail	The camera produces blurry, distorted, or low-quality images or fails to function.	Camera: Canon	06KL	5.45
4765	W-Fi Chipsets	Structure Integrity: Checking if parts have the right structure	Fail	Wi-Fi chipset fails to connect, drops connections, or delivers slow speeds.	Wi-Fi Chipsets: Qualcomm	08OP	9.78
5628	Processor CPU	Structure Integrity: Checking if parts have the right structure	Fail	The laptop's CPU underperforms, overheats, or crashes during benchmark tests.	Processor CPU: Intel	11UV	5.67
1275	Display	Structure Integrity: Checking if parts have the right structure	Fail	The laptop's display has dead pixels, flickers, or displays distorted images.	Display: Sharp	14AB	7.99
3891	Battery	Structure Integrity: Checking if parts have the right structure	Fail	The laptop's battery capacity significantly degrades, leading to short usage times or failure to hold a charge	Battery: Sony Energy	15CD	5.36
6950	Storage SSD	Structure Integrity: Checking if parts have the right structure	Fail	Structural damage in the SSD results in performance issues or failure.	Memory RAM: Team Group	22QR	8.79
2386	Camera	Structure Integrity: Checking if parts have the right structure	Fail	Structural damage in the camera results in blurry or distorted images or camera malfunction.	Battery: Sanyo	25WX	7.45
7254	Display	Structure Integrity: Checking if parts have the right structure	Fail	Watch display structural damage in the display results in dead pixels, flickering, or distorted images.	Memory RAM: Patriot	32KL	8.45
5417	Battery	Structure Integrity: Checking if parts have the right structure	Fail	Watch battery has structural damage, leads to a capacity decrease with inability to hold a charge.	Storage SSD: Toshiba	33MN	5.10

11 rows in set (0.001 sec)

05 Retrieve Cost and Manufacturing Data:

```
MariaDB [spm]> SELECT
    ->     manufacturer,
    ->     AVG(manufacturing_cost) AS avg_manufacturing_cost
    -> FROM
    ->     quality_manufacturing
    -> GROUP BY
    ->     manufacturer;
```

manufacturer	avg_manufacturing_cost
Audio Components: Beats by Dre	8.010000
Audio Components: JBL	9.230000
Audio Components: Sennheiser	6.890000
Battery: Amperex Technology	6.340000
Battery: LG Chem	7.920000
Battery: Sanyo	7.450000

This query calculates the average manufacturing cost for each manufacturer, aiding in the identification of manufacturers with lower average costs, which can contribute to cost optimization in manufacturing.

06 Merging Tables for Future Calculation.

```
MariaDB [spm]> create table manufacturer_cost_and_quantity as
    -> select a.manufacturer_id, a.manufacturer, a.component_id, b.manufacturer_cost, b.quantity
    -> from manufacturer_updated a
    -> join manufacturer_cost_updated b
    -> on a.manufacturer_id = b.manufacturer_id;
Query OK, 36 rows affected (0.024 sec)
Records: 36  Duplicates: 0  Warnings: 0

MariaDB [spm]> select * from manufacturer_cost_and_quantity;
+-----------------+-------------------+--------------+-------------------+----------+
| manufacturer_id | manufacturer      | component_id | manufacturer_cost | quantity |
+-----------------+-------------------+--------------+-------------------+----------+
| 01AB            | AMD               |         2387 |              6.24 |      100 |
| 02CD            | Corsair           |         5129 |              8.53 |      100 |
| 03EF            | Western Digital   |         7642 |              5.78 |      100 |
| 04GH            | AU Optronics      |         3498 |              9.01 |      100 |
| 05IJ            | LG Chem           |         6154 |              7.92 |      100 |
| 06KL            | Canon             |         9270 |              5.45 |      100 |
```

Merging the tables manufacturer and manufacturer_cost into manufacturer_cost_and_quantity to bring manufacturer and cost information in one table for future calculation. Using Inner join.

Identifying Rate or Failure and Total Cost

07

```
MariaDB [spm]> CREATE TABLE component_costs AS
    -> SELECT
    ->   r.component_id,
    ->   r.pass_count,
    ->   r.fail_count,
    ->   r.total_test_numbers,
    ->   r.pass_rate,
    ->   r.fail_rate,
    ->   c.manufacturer as manufacturer_name,
    ->   c.manufacturer_cost,
    ->   c.quantity,
    ->   c.manufacturer_cost * c.quantity AS total_cost
    -> FROM (
    ->   SELECT
    ->     component_id,
    ->     SUM(CASE WHEN fail_present = 0 THEN 1 ELSE 0 END) AS pass_count,
    ->     SUM(CASE WHEN fail_present > 0 THEN 1 ELSE 0 END) AS fail_count,
    ->     COUNT(*) AS total_test_numbers,
    ->     SUM(CASE WHEN fail_present = 0 THEN 1 ELSE 0 END) / COUNT(*) * 100 AS pass_rate,
    ->     SUM(CASE WHEN fail_present > 0 THEN 1 ELSE 0 END) / COUNT(*) * 100 AS fail_rate
    ->   FROM (
    ->     SELECT
    ->       component_id,
    ->       test_number,
    ->       SUM(CASE WHEN Is_Pass_Fail = 'Fail' THEN 1 ELSE 0 END) AS fail_present
    ->     FROM
    ->       QC
    ->     GROUP BY
    ->       component_id,
    ->       test_number
    ->   ) AS subquery
    ->   GROUP BY
    ->     component_id
    -> ) r
    -> JOIN manufacturer_cost_and_quantity c ON r.component_id = c.component_id;
Query OK, 5 rows affected (0.043 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

The component_costs table is like a detailed report card for each component. It tells us how many times each component passed or failed tests, how many times it was tested, and all the costs involved. We get this information by combining data from the QC and manufacturer_cost_and_quantity tables. It gives us a really good picture of how well the parts perform and how much they cost.

```
MariaDB [spm]> select * from component_costs;
```

component_id	pass_count	fail_count	total_test_numbers	pass_rate	fail_rate	manufacturer_name	manufacturer_cost	quantity	total_cost
1836	82	18	100	82.0000	18.0000	Sennheiser	6.89	100	689.00
1275	98	2	100	98.0000	2.0000	Sharp	7.99	100	799.00
1049	95	5	100	95.0000	5.0000	BOE Technology	6.56	100	656.00
2386	78	22	100	78.0000	22.0000	Sanyo	7.45	100	745.00
1248	56	44	100	56.0000	44.0000	Cooler Master	9.12	100	912.00

```
5 rows in set (0.001 sec)
```

Final Component Analysis

```
MariaDB [spm]> CREATE TABLE final_component_analysis AS
    -> SELECT
    ->   component_id,
    ->   pass_count,
    ->   fail_count,
    ->   total_test_numbers,
    ->   pass_rate as pass_rate_in_percentage,
    ->   fail_rate as fail_rate_in_percentage,
    ->   manufacturer_name,
    ->   manufacturer_cost as manufacturer_cost_per_component,
    ->   quantity as total_ordered_quantity,
    ->   total_cost,
    ->   -- FlagLevel based on pass_rate
    ->   CASE
    ->     WHEN pass_rate > 90 THEN 'green'
    ->     WHEN pass_rate BETWEEN 80 AND 90 THEN 'yellow'
    ->     ELSE 'red'
    ->   END AS FlagLevel,
    ->   -- Calculate the AmountInDollarsForFailTest
    ->   total_cost * (fail_rate / 100) AS Amount_in_dollars_for_fail_test,
    ->   -- MarginalValue by the deal with the manufacturer
    ->   total_cost * 0.08 AS MarginalValue,
    ->   -- Calculate refund_amount_for_fail_test
    ->   (total_cost * (fail_rate / 100)) - (total_cost * 0.08) AS refund_amount_for_fail_test,
    ->   -- Calculate LossAmount
    ->   (total_cost * (pass_rate / 100)) - (total_cost * .92) AS LossAmount,
    ->    ((total_cost * (pass_rate / 100)) - (total_cost * .92)) * 100 / total_cost  as Loss_in_percentage
    -> FROM
    ->   component_costs;
Query OK, 5 rows affected (0.027 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [spm]> select * from  final_component_analysis
    -> \cs
    -> \c
MariaDB [spm]> select * from  final_component_analysis;
```

Selecting Specific Columns from the original table.

Flagging Component Performance: Based on the pass rate of each component. If the rate is above 90%, it gets a 'green' flag, between 80% and 90% gets a 'yellow' flag, and anything below 80% gets a 'red' flag.

Calculating Financial Metrics: Such as the cost of failed tests, a marginal value of 8% of the total from the manufacturer, and the potential refund amount for failed tests units.

Determining Losses: It calculates the loss due to failed tests and represents this loss as a percentage of the total cost.

component_id	pass_count	fail_count	total_test_numbers	pass_rate_in_percentage	fail_rate_in_percentage	manufacturer_name	manufacturer_cost_per_component	total_ordered_quantity	total_cost	FlagLevel	Amount_in_dollars_for_fail_test	MarginalValue	refund_amount_for_fail_test	LossAmount	Loss_in_percentage
1836	82	18	100	82.0000	18.0000	Sennheiser	6.89	100	689.00	yellow	124.0200000000	55.1200	68.9000000000	-68.9000000000	-10.00000000000000
1275	98	2	100	98.0000	2.0000	Sharp	7.99	100	799.00	green	15.9800000000	63.9200	-47.9400000000	47.9400000000	6.00000000000000
1049	95	5	100	95.0000	5.0000	BOE Technology	6.56	100	656.00	green	32.8000000000	52.4800	-19.6800000000	19.6800000000	3.00000000000000
2386	78	22	100	78.0000	22.0000	Sanyo	7.45	100	745.00	red	163.9000000000	59.6000	104.3000000000	-104.3000000000	-14.00000000000000
1248	56	44	100	56.0000	44.0000	Cooler Master	9.12	100	912.00	red	401.2800000000	72.9600	328.3200000000	-328.3200000000	-36.00000000000000

Simplification of the Previous Table for a Better Understanding

```
MariaDB [spm]> create table final_result as select component_id as comp_id
    -> ,pass_count as pass_cnt
    -> ,fail_count as fail_cnt
    -> ,total_test_numbers as total_tests
    -> ,round(pass_rate_in_percentage, 2) as pass_rate_pct
    -> ,round(fail_rate_in_percentage, 2) as fail_rate_pct
    -> ,manufacturer_name as manuf_name
    -> ,manufacturer_cost_per_component as cost_per_comp
    -> ,total_ordered_quantity as ordered_qty
    -> ,total_cost as total_cost
    -> ,FlagLevel as flag_lvl
    -> ,round(Amount_in_dollars_for_fail_test, 2) as fail_test_cost
    -> ,round(MarginalValue, 2) as marginal_val
    -> ,round(case when Amount_in_dollars_for_fail_test > MarginalValue then (Amount_in_dollars_for_fail_test - marginalvalue) else 0 end , 2) as refund_fail_test
    -> ,round(case when Amount_in_dollars_for_fail_test > marginalvalue then (marginalvalue) else Amount_in_dollars_for_fail_test end, 2) as loss_amt
    -> ,round((case when Amount_in_dollars_for_fail_test > marginalvalue then (Amount_in_dollars_for_fail_test - marginalvalue) else Amount_in_dollars_for_fail_test end *100 / total_cost ), 2) as loss_pct
    -> from final_component_analysis;
Query OK, 5 rows affected (0.026 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [spm]> select * from final_results;
ERROR 1146 (42S02): Table 'spm.final_results' doesn't exist
MariaDB [spm]> select * from final_result;
```

comp_id	pass_cnt	fail_cnt	total_tests	pass_rate_pct	fail_rate_pct	manuf_name	cost_per_comp	ordered_qty	total_cost	flag_lvl	fail_test_cost	marginal_val	refund_fail_test	loss_amt	loss_pct
1836	82	18	100	82.00	18.00	Sennheiser	6.89	100	689.00	yellow	124.02	55.12	68.90	55.12	10.00
1275	98	2	100	98.00	2.00	Sharp	7.99	100	799.00	green	15.98	63.92	0.00	15.98	2.00
1049	95	5	100	95.00	5.00	BOE Technology	6.56	100	656.00	green	32.80	52.48	0.00	32.80	5.00
2386	78	22	100	78.00	22.00	Sanyo	7.45	100	745.00	red	163.90	59.60	104.30	59.60	14.00
1248	56	44	100	56.00	44.00	Cooler Master	9.12	100	912.00	red	401.28	72.96	328.32	72.96	36.00

```
5 rows in set (0.001 sec)
```

The following query simplifies the name of the columns and rounds up the integers to 2 decimal places for a cleaner view and a better understanding.

Final thoughts

If we pick the red flag manufacturer, Cooler Master, our analysis shows a loss of $72.96 per 100 components. Scaling this to a more realistic volume, such as 10 million components, projects a staggering loss of $7.3 million.

7296000+

Dollars

To address this issue, several options can be explored, including:

- Cancelling the deal and seeking a more reliable company.
- Reassessing the marginal value to potentially increase the percentage in order to mitigate the losses.

Do you have any questions?

support@techgear.com
+13478910234
Techgear.inc

THANKS



Credit for this presentation goes to Sergiu. Various sources contributed to its creation, including but not limited to ChatGPT, Google, and StackOverflow.