BY JATIN BAGORIA

TESLACAR

MANUFACTURING AND QUALITY CONTROL ANALYSIS

HOME SERVICE ABOUT US CONTACT US

WELCOME TO TESLA ANALYSIS

IDENTIFYING ANOMALIES IN SHIFT PERFORMANCE

```sql
WITH ShiftPerformance AS (
    SELECT shift,
           AVG(quantity_produced) AS avg_output,
           STDDEV(quantity_produced) AS std_dev
    FROM Production
    GROUP BY shift
)
SELECT shift, avg_output, std_dev
FROM ShiftPerformance
WHERE std_dev > (SELECT AVG(std_dev) FROM ShiftPerformance);
```

	Shift	Avg_Output	Std_Dev_Output
▶	Morning	10200.00	2500.00
	Night	9800.00	4000.00

PREDICTING DEFECTS USING TRENDS

```sql
SELECT DATE_TRUNC('week', date_detected) AS week,
defect_type,
COUNT(defect_id) AS defect_count,
LAG(COUNT(defect_id), 1)
OVER (PARTITION BY defect_type
ORDER BY DATE_TRUNC('week', date_detected)) AS prev_week_count
FROM Defects
GROUP BY week, defect_type
HAVING COUNT(defect_id) > 1.2 * COALESCE(prev_week_count, 0);
```

	Week	Defect_Type	Defect_Count	Prev_Week_Count
▶	Week 1	Alignment	12	NULL
	Week 2	Alignment	15	12
	Week 1	Overheating	8	NULL

OPTIMAL MAINTENANCE SCHEDULING

```sql
WITH FailureProbability AS (
SELECT machine_id,
AVG(CASE WHEN defect_id IS NOT NULL THEN 1 ELSE 0 END) AS defect_probability
FROM Production
LEFT JOIN Defects USING (batch_id)
GROUP BY machine_id
)
SELECT machine_id, defect_probability
FROM FailureProbability
ORDER BY defect_probability DESC
LIMIT 5;
```

machine_id	defect_probability
Machine_1	0.85
Machine_3	0.78
Machine_7	0.75
Machine_4	0.70
Machine_5	0.68

BOTTLENECK DETECTION

```sql
SELECT production_line_id,
AVG(processing_time) AS avg_processing_time,
 PERCENT_RANK() OVER (ORDER BY AVG(processing_time) DESC) AS bottleneck_rank
FROM Production
GROUP BY production_line_id
HAVING bottleneck_rank <= 0.2;
```

production_line_id	avg_processing_time	bottleneck_rank
Line_3	120	0.20
Line_1	115	0.18
Line_5	110	0.15
Line_8	105	0.12
Line_7	100	0.10

ROOT CAUSE ANALYSIS FOR DEFECTS

```sql
SELECT machine_id, defect_type, shift,
       COUNT(defect_id) AS defect_count,
       AVG(severity) AS avg_severity
FROM Defects
JOIN Production USING (batch_id)
WHERE severity > 3
GROUP BY machine_id, defect_type, shift
ORDER BY defect_count DESC, avg_severity DESC;
```

machine_id	defect_type	shift	defect_count	avg_severity
Machine_1	Alignment	Morning	15	4.5
Machine_3	Overheating	Night	12	4.7
Machine_2	Leakage	Morning	10	4.3
Machine_4	Misalignment	Afternoon	9	4.6
Machine_5	Crack	Night	8	4.2

EFFICIENCY ANALYSIS BY MATERIAL USED

```sql
SELECT material_id,
       AVG(quantity_produced) AS avg_output,
       COUNT(defect_id) AS defect_count,
       AVG(defect_severity) AS avg_severity
FROM Production
LEFT JOIN Defects USING (batch_id)
GROUP BY material_id
ORDER BY avg_output DESC;
```

material_id	avg_output	defect_count	avg_severity
Material_1	2000	15	4.3
Material_2	1800	10	3.9
Material_3	1700	5	4.1
Material_4	1600	8	4.0
Material_5	1500	12	4.2

CORRELATION BETWEEN OPERATOR AND DEFECT RATES

```sql
SELECT operator_id,
       COUNT(DISTINCT defect_id) * 1.0 / COUNT(batch_id) AS defect_rate,
       AVG(quantity_produced) AS avg_output
FROM Production
LEFT JOIN Defects USING (batch_id)
GROUP BY operator_id
HAVING defect_rate > 0.05
ORDER BY defect_rate DESC;
```

	operator_id	defect_rate	avg_output
▶	Operator_1	0.08	1500
	Operator_3	0.12	1400
	Operator_5	0.07	1300
	Operator_2	0.09	1350
	Operator_4	0.06	1250

DETECTING SEASONAL DEFECT PATTERNS

```sql
SELECT TO_CHAR(date_detected, 'Month') AS month,
       defect_type,
       COUNT(defect_id) AS total_defects
FROM Defects
GROUP BY month, defect_type
ORDER BY total_defects DESC;
```

month	defect_type	total_defects
January	Leakage	25
February	Overheating	20
March	Alignment	18
April	Crack	15
May	Misalignment	12

RANKING PRODUCT LINES BY PROFITABILITY

```sql
SELECT product_line,
       SUM(production_cost) AS total_cost,
       SUM(revenue) AS total_revenue,
       (SUM(revenue) - SUM(production_cost)) AS net_profit
FROM Production
GROUP BY product_line
ORDER BY net_profit DESC;
```

	product_line	total_cost	total_revenue	net_profit
▶	Product Line 1	50000	80000	30000
	Product Line 2	45000	70000	25000
	Product Line 3	40000	65000	25000
	Product Line 4	35000	60000	25000
	Product Line 5	30000	50000	20000

REAL-TIME QUALITY CONTROL ALERTS

```sql
SELECT batch_id, defect_type, severity, inspection_time
FROM Defects
WHERE inspection_time > NOW() - INTERVAL '1 hour'
ORDER BY severity DESC;
```

batch_id	defect_type	severity	inspection_time
Batch_101	Leakage	5	2024-11-20 14:45:00
Batch_102	Overheating	4	2024-11-20 14:40:00
Batch_103	Crack	3	2024-11-20 14:30:00
Batch_104	Misalignment	2	2024-11-20 14:25:00
Batch_105	Misalignment	1	2024-11-20 14:15:00

CUMULATIVE DEFECT RATE OVER TIME

```sql
SELECT DATE_TRUNC('month', date_detected) AS month,
       COUNT(defect_id) * 1.0 / COUNT(DISTINCT batch_id) AS defect_rate
FROM Defects
JOIN Production USING (batch_id)
GROUP BY month
ORDER BY month;
```

month	defect_rate
2024-01-01	0.08
2024-02-01	0.06
2024-03-01	0.07
2024-04-01	0.05
2024-05-01	0.09

PRODUCTION FORECASTING

```sql
SELECT DATE_TRUNC('month', production_date) AS month,
       SUM(quantity_produced) AS total_output,
       AVG(quantity_produced) AS avg_output
FROM Production
GROUP BY month
ORDER BY month;
```

month	total_output	avg_output
2024-01-01	50000	1600
2024-02-01	48000	1600
2024-03-01	51000	1700
2024-04-01	47000	1500
2024-05-01	49000	1600

ANALYSIS OF FAILED MAINTENANCE EFFORTS

```sql
SELECT maintenance_id,
       COUNT(defect_id) AS defects_post_maintenance
FROM Maintenance
LEFT JOIN Defects ON Maintenance.machine_id = Defects.machine_id
        AND Defects.date_detected > Maintenance.maintenance_date
GROUP BY maintenance_id
HAVING defects_post_maintenance > 5;
```

maintenance_id	defects_post_maintenance
Maintenance_101	8
Maintenance_102	10
Maintenance_103	7
Maintenance_104	9
Maintenance_105	6

MULTI-DIMENSIONAL ANALYSIS OF PRODUCTION ISSUES

```sql
SELECT shift, machine_id, operator_id,
       COUNT(defect_id) AS defect_count,
       AVG(processing_time) AS avg_processing_time
FROM Production
LEFT JOIN Defects USING (batch_id)
GROUP BY shift, machine_id, operator_id
ORDER BY defect_count DESC;
```

shift	machine_id	operator_id	defect_count	avg_processing_time
Shift_1	Machine_1	Operator_1	12	15.2
Shift_1	Machine_2	Operator_2	10	14.8
Shift_2	Machine_1	Operator_3	8	13.5
Shift_2	Machine_3	Operator_1	7	12.3
Shift_3	Machine_4	Operator_4	6	16.0

COST ANALYSIS FOR DEFECT MANAGEMENT

```sql
SELECT defect_type,
       SUM(repair_cost) AS total_repair_cost,
       AVG(repair_cost) AS avg_repair_cost
FROM Defects
GROUP BY defect_type
ORDER BY total_repair_cost DESC;
```

	defect_type	total_repair_cost	avg_repair_cost
▶	Leakage	15000	500
	Overheating	12000	400
	Misalignment	10000	350
	Crack	8000	400
	Alignment	5000	350

BY JATIN BAGORIA

THANK YOU