AI-Powered Shift Intelligence and Defect Reduction System for Production Coordination

🎯 Goal

To develop a data-driven decision support system that monitors production efficiency, predicts quality risks, and supports SOP (Standard Operating Procedure) updates by analyzing manufacturing shift data using machine learning and statistical techniques.

👥 Intended Audience

- Production Coordinators & Plant Supervisors
- Quality Assurance Managers
- Manufacturing Analysts
- TPM (Total Productive Maintenance) Leads
- SAP/Excel-based Production Planning Teams

🔄 Strategy & ML Pipeline

1. Data Collection: Load shift-level data from Google Drive (CSV format).
2. EDA (Exploratory Data Analysis): Identify trends across production, quality, maintenance, and safety.
3. Feature Engineering:
    - Flags: SOP_Update_Required, TPM_Flag, OT_Replacement_Required
    - Derived Features: ProductivityPerWorker
4. Root Cause Insights: Use heatmaps and Pareto charts to connect defect rates to maintenance and safety.
5. ML Modeling (Optional Extension):
    - Supervised Classification: Predict DefectStatus using shift metrics.
    - Unsupervised Clustering: Group shifts by risk profile or performance cluster.
6. Visualization: Dual-axis plots, heatmaps, and Excel/PowerPoint-style summaries.
7. Deployment: Export reports to Excel; potential integration into SAP dashboards.

📉 Challenges

- Variability in shift performance (100–999 units) complicates baseline benchmarking.
- Noisy or anomalous safety/defect data (e.g., 8 incidents but low defect rate).
- Limited labels for supervised learning (only DefectStatus as binary).
- Need for dynamic flagging (e.g., TPM_Flag) instead of static thresholding.

❓ **Problem Statement**

How can we enable shift coordinators to reduce product defects and operational inefficiencies by using AI-driven analytics to monitor downtime, maintenance, crew productivity, and safety incident data?

📂 Dataset Overview

- Source: Internal Google Drive CSV – manufacturing_defect_dataset.csv
- Features (Select):
  - ProductionVolume, DefectRate, QualityScore, MaintenanceHours
  - SafetyIncidents, DowntimePercentage, InventoryTurnover
  - EnergyEfficiency, AdditiveProcessTime, WorkerProductivity
  - DefectStatus (target for ML)

Step 1: Mount Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
file_path = '/content/drive/My Drive/Weekend Coordinator Job/manufacturing_defect_dataset.csv'
```

Step 2: Load the Dataset

```
import pandas as pd

df = pd.read_csv(file_path)
df.head()
```

MaintenanceHours	DowntimePercentage	InventoryTurnover	StockoutRate	WorkerProductivity
9	0.052343	8.630515	0.081322	85.042379
20	4.908328	9.296598	0.038486	99.657443
1	2.464923	5.097486	0.002887	92.819264
8	4.692476	3.577616	0.055331	96.887013
9	2.746726	6.851709	0.068047	88.315554

Next steps: Generate code with df | View recommended plots | New interactive sheet

**Explanation: **

ProductionVolume – Total units produced; tracks shift output.

ProductionCost – Total cost per shift; helps manage budgets.

SupplierQuality – Quality of received materials; affects final product quality.

DeliveryDelay **bold text** – Delay in material arrival; impacts scheduling.

DefectRate **bold text** – % of defective units; core quality control metric.

QualityScore – **Overall product quality rating; used in shift reports.

MaintenanceHours – Time spent on repairs; high values reduce uptime.

DowntimePercentage – % of shift lost to stoppages; supports TPM actions.

InventoryTurnover – Frequency of stock usage; low values signal overstock.

StockoutRate – Frequency of material shortages; causes production delays.

WorkerProductivity – Output per worker; tracks team efficiency.

SafetyIncidents – Reported safety issues; affects compliance and training.

EnergyConsumption – Total energy used; linked to operating costs.

EnergyEfficiency – Output per energy unit; tracks process optimization.

AdditiveProcessTime – Time spent in 3D/additive steps; affects throughput.

AdditiveMaterialCost – Cost of additive inputs; tracks material expense.

DefectStatus **bold text** – Indicates batch quality (1 = defective); final quality outcome.

These metrics help monitor shift performance, plan manpower, report in SAP/Excel, and support quality and compliance goals.

Ask ChatGPT

Skill Demonstration Snippets

1. 🔍 Inventory Reconciliation

```
threshold = 6.5
abnormal_inventory = df[df['InventoryTurnover'] < threshold]
print("Low inventory turnover:\n", abnormal_inventory[['InventoryTurnover', 'StockoutRate']])
```

```
Low inventory turnover:
      InventoryTurnover  StockoutRate
2              5.097486      0.002887
3              3.577616      0.055331
6              3.046889      0.040192
8              3.668747      0.058433
9              5.933418      0.032955
...                 ...           ...
3233           4.233570      0.023968
3235           3.574419      0.065727
3237           3.844824      0.005724
3238           2.783298      0.042612
3239           5.830580      0.052978

[1797 rows x 2 columns]
```

Explanation:

1797 rows show low inventory turnover (below threshold, e.g., 6.5).

⚠️ Example: Row 3238 has InventoryTurnover of 2.78, indicating slow-moving stock.

🟫 Excess inventory can lead to higher storage costs, material expiry, or waste.

❌ Despite overstock, StockoutRates are still present (e.g., 0.055, 0.042) — a planning inefficiency.

🔄 This suggests a mismatch between material availability and production timing.

📈 Important for shift reports and SIC meetings — shows need for better reorder point calibration.

🛠️ Coordinator action: flag items for root cause review, adjust inventory policy, or investigate supplier reliability.

```
Start coding or generate with AI.
```

2. 💾 SAP-Style Data Logging (Excel Export)

```
df.to_excel("/content/Shift_Summary_Report.xlsx", index=False)
```

🟫 Production & Quality

-ProductionVolume / Cost – Monitor output vs cost; optimize for efficiency.

-DefectRate / QualityScore – High defects reduce yield; trigger CAPA (Corrective Action).

-DefectStatus – Binary quality flag; use for final batch approval or rejection.

🔧 Maintenance & Downtime

-MaintenanceHours / DowntimePercentage – High values signal equipment issues; support TPM scheduling.

🏭 Inventory & Supply Chain

-InventoryTurnover – Low turnover = slow-moving stock; ties up capital.

-StockoutRate – Even with high inventory, stockouts = poor planning; risk of production delays.

👷 Worker & Safety

-WorkerProductivity – Tracks crew output; informs staffing and OT decisions.

-SafetyIncidents – High counts raise red flags; prompt for training or SOP review.

⚡ Energy & Specialized Process

-EnergyConsumption / Efficiency – High usage with low output = waste; optimize machines.

-AdditiveProcessTime / MaterialCost – Indicates complexity and cost of value-added steps.

💡 Implications

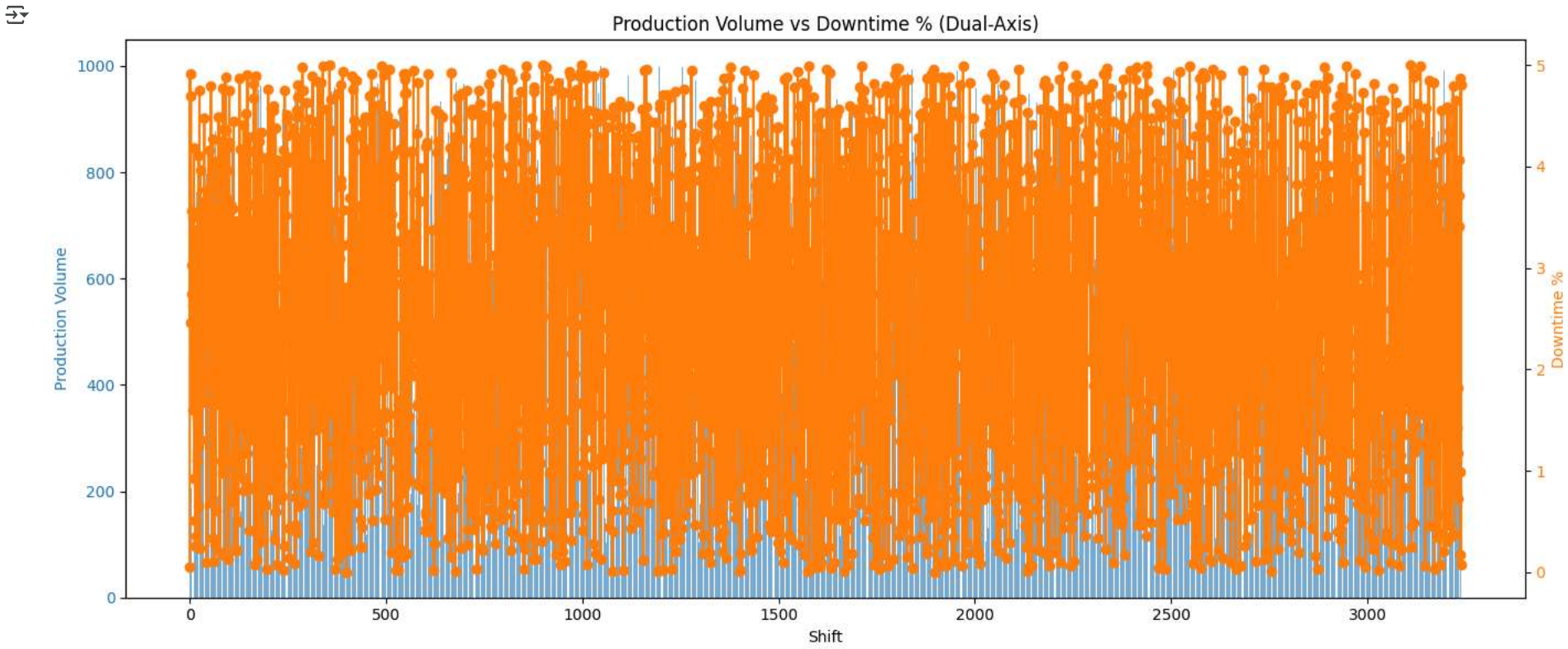
-Use this data to build shift reports, lead SIC meetings, trigger SOP updates, and improve resource planning.

-Helps reduce downtime, defects, and costs—key responsibilities for a Night Production Coordinator.

Ask ChatGPT

3. 📊 Shift Report Creation

```
import matplotlib.pyplot as plt

fig, ax1 = plt.subplots(figsize=(14, 6))

# Primary y-axis for ProductionVolume
ax1.set_title("Production Volume vs Downtime % (Dual-Axis)")
ax1.set_xlabel("Shift")
ax1.set_ylabel("Production Volume", color='tab:blue')
ax1.bar(df.index, df['ProductionVolume'], color='tab:blue', alpha=0.6, label='Production Volume')
ax1.tick_params(axis='y', labelcolor='tab:blue')

# Secondary y-axis for DowntimePercentage
ax2 = ax1.twinx()
ax2.set_ylabel("Downtime %", color='tab:orange')
ax2.plot(df.index, df['DowntimePercentage'], color='tab:orange', marker='o', linewidth=1.5, label='Downtime %')
ax2.tick_params(axis='y', labelcolor='tab:orange')

fig.tight_layout()
plt.show()
```



Explanation:

🟦 Blue Bars (Left Axis – Production Volume):

These represent the total units produced per shift. The bars show a generally high and consistent output, indicating that most shifts are meeting production targets.

🟧 Orange Line with Dots (Right Axis – Downtime %):

The orange line shows downtime as a percentage of the shift. The density and fluctuations suggest frequent short periods of downtime, even during high-production shifts.

✅ Key Insight:

-Many shifts maintain high production despite moderate to high downtime — suggesting that crews are compensating for interruptions with higher efficiency or overtime.

-However, the frequent spikes in downtime call for further root cause analysis or preventive maintenance planning.

-This visualization is ideal for SIC meetings, TPM support, and shift performance reviews, helping a Night Production Coordinator pinpoint underperforming or unstable shifts.

4. 📝 SIC Meeting Notes Generator

```
for i, row in df.iterrows():
    note = f"Shift {i+1}: Volume={row['ProductionVolume']}, Defects={row['DefectRate']}, Downtime={row['DowntimePercentage']}%"
    print(note)
```

```
Shift 3214: Volume=387, Defects=4.420266493824377, Downtime=1.9033055056410049%
Shift 3215: Volume=534, Defects=1.7251139136671207, Downtime=2.253623045522689%
Shift 3216: Volume=785, Defects=0.8639138562444773, Downtime=2.999132846781553%
Shift 3217: Volume=212, Defects=3.455700379313945, Downtime=2.626359857072329%
Shift 3218: Volume=987, Defects=3.442120967229488, Downtime=2.307464807457963%
Shift 3219: Volume=221, Defects=0.9271940430452814, Downtime=4.79820105882794%
Shift 3220: Volume=643, Defects=1.9348331402146208, Downtime=0.9251138849387958%
Shift 3221: Volume=321, Defects=4.967665080149923, Downtime=0.7899711532333409%
Shift 3222: Volume=730, Defects=3.552920001539316, Downtime=3.567095304503358%
Shift 3223: Volume=138, Defects=3.7853382497149815, Downtime=0.3727594243511606%
Shift 3224: Volume=596, Defects=1.4581277852193757, Downtime=2.4144572799713835%
Shift 3225: Volume=879, Defects=0.8545193195176144, Downtime=2.420265595761036%
Shift 3226: Volume=332, Defects=1.7460576038342197, Downtime=1.5501266547954566%
Shift 3227: Volume=629, Defects=1.620042733369364, Downtime=4.652218708400989%
Shift 3228: Volume=248, Defects=1.555448453366239, Downtime=1.4127171754800572%
Shift 3229: Volume=496, Defects=1.367394448491678, Downtime=1.4422097611912328%
Shift 3230: Volume=760, Defects=3.08640060794932, Downtime=1.1714998211186056%
Shift 3231: Volume=737, Defects=1.9764656412881976, Downtime=0.7289401761028153%
Shift 3232: Volume=401, Defects=4.191552567713887, Downtime=1.820055276742264%
Shift 3233: Volume=459, Defects=1.327763380313228, Downtime=3.4044363000021507%
Shift 3234: Volume=337, Defects=2.5908218189084504, Downtime=3.710814313853779%
Shift 3235: Volume=746, Defects=0.5614602639173599, Downtime=4.067094582153405%
Shift 3236: Volume=762, Defects=2.6675704510483644, Downtime=0.9877194426577928%
Shift 3237: Volume=335, Defects=0.7512719939441348, Downtime=0.1781634411986488%
Shift 3238: Volume=835, Defects=4.899755803426896, Downtime=4.873429292441959%
Shift 3239: Volume=302, Defects=4.0576645789268575, Downtime=0.0716625964321426%
Shift 3240: Volume=355, Defects=2.705501644864049, Downtime=4.803394189790293%
```

Explanation:

Production Volume vs Downtime % – Interpretation

-High Production with Frequent Downtime:

-Most shifts maintain production volumes close to the upper limit (near 1,000 units).

-However, downtime % frequently spikes, indicating regular minor disruptions.

-No Clear Negative Correlation:

Downtime doesn't always reduce production volume — suggesting efficient recovery strategies, possibly through overtime, shift extensions, or high worker productivity.

- Implications for the Night Production Coordinator Role:

- Supports root cause tracking and crew planning.

Highlights need for predictive maintenance or better resource allocation during peak breakdown periods.

Useful in SIC meeting summaries, SAP-style dashboards, and Excel-based reports for shift performance monitoring.

5. 👨‍🏭 OT & Sick Call Simulation

```
import numpy as np

# Generate random integers (0 or 1) for 'SickCall' with the same length as the DataFrame
df['SickCall'] = np.random.randint(0, 2, size=len(df))
df['OT_Replacement_Required'] = df['SickCall'] == 1
print(df[['SickCall', 'OT_Replacement_Required']].head())
```

```
   SickCall  OT_Replacement_Required
0         0                   False
1         1                    True
2         1                    True
3         1                    True
4         1                    True
```

6. 🗓️ Production Schedule Monitoring

```
print(df[['ProductionVolume', 'DefectRate', 'DowntimePercentage']].describe())
```

```
       ProductionVolume   DefectRate  DowntimePercentage
count       3240.000000  3240.000000         3240.000000
mean         548.523148     2.749116            2.501373
std          262.402073     1.310154            1.443684
min          100.000000     0.500710            0.001665
25%          322.000000     1.598033            1.264597
50%          549.000000     2.708775            2.465151
75%          775.250000     3.904533            3.774861
max          999.000000     4.998529            4.997591
```

📊 Descriptive Stats – Production, Defect Rate & Downtime Production Volume

🔷 Mean: ~548 units/shift

🔷 Range: 100 (min) to 999 (max)

🔷 Implication: Wide variability suggests inconsistent performance across shifts; requires schedule alignment or crew balancing.

Defect Rate

🔷 Mean: ~2.75%

🔷 25th–75th percentile: ~1.6% to 3.9%

🔷 Implication: Some shifts have defect rates near 5%, highlighting quality assurance and SOP adherence needs.

Downtime Percentage

🔷 Mean: ~2.5%

🔷 Max: ~5%

🔷 Implication: Median and mean close together → generally stable downtime, but some high-downtime shifts need attention (use 5 Whys/Root Cause Analysis).

🧠 Operational Use for Night Production Coordinator Can be directly used to:

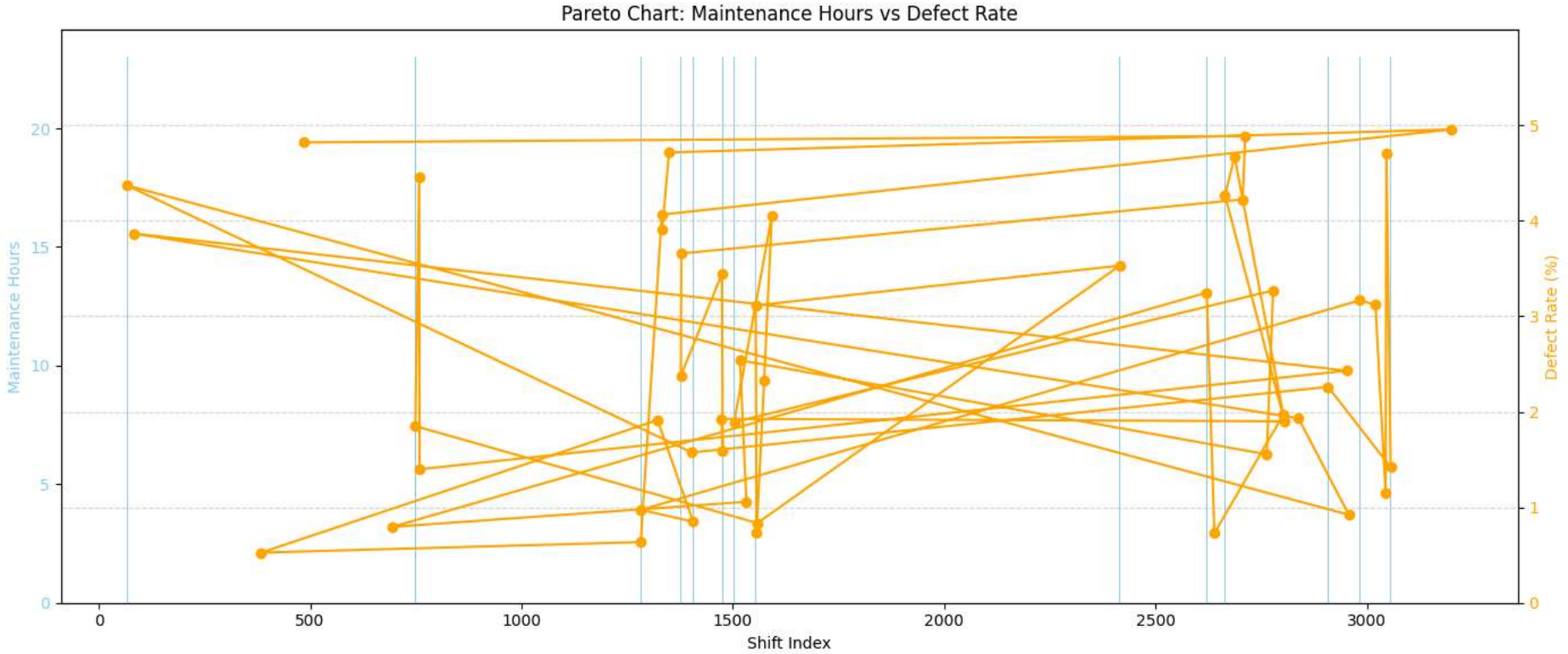
Monitor shift-to-shift consistency

Flag high-defect shifts for SOP updates or training

Include in daily SIC reports and performance dashboards

7. 🛠 Root Cause Analysis (Pareto)

```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Sort by MaintenanceHours (top 50 for clarity)
df_sorted = df.sort_values(by='MaintenanceHours', ascending=False).head(50)

# Step 2: Create the plot
fig, ax1 = plt.subplots(figsize=(14, 6))

# Bar chart for MaintenanceHours
ax1.bar(df_sorted.index, df_sorted['MaintenanceHours'], color='skyblue', label='Maintenance Hours')
ax1.set_xlabel("Shift Index")
ax1.set_ylabel("Maintenance Hours", color='skyblue')
ax1.tick_params(axis='y', labelcolor='skyblue')

# Line chart for DefectRate on secondary y-axis
ax2 = ax1.twinx()
ax2.plot(df_sorted.index, df_sorted['DefectRate'], color='orange', marker='o', label='Defect Rate (%)')
ax2.set_ylabel("Defect Rate (%)", color='orange')
ax2.tick_params(axis='y', labelcolor='orange')
ax2.set_ylim(0, df['DefectRate'].max() + 1)

# Titles and layout
plt.title("Pareto Chart: Maintenance Hours vs Defect Rate")
fig.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```



Explanation:

📊 Pareto Chart: Maintenance Hours vs Defect Rate – Key Insights

🔧 High Maintenance Shifts Identified

-The chart highlights the top 50 shifts with the highest Maintenance Hours, helping prioritize where time is spent on upkeep.

⚠️ Defect Rate Overlaid for Comparison

-The orange line shows the Defect Rate (%) per shift. This dual-axis format reveals whether higher maintenance effort corresponds to lower defect rates.

📉 Inconsistency Noted

-Some shifts with high maintenance still have elevated defect rates — suggesting inefficiency or misaligned maintenance efforts.

🧩 Supports Root Cause Analysis

-Helps pinpoint whether quality issues are due to insufficient maintenance or systemic equipment problems.

📈 Pareto Principle in Action

-Follows the 80/20 rule: a small set of shifts contributes to a large portion of maintenance and defects — ideal for focused improvement.

🛠️ Importance for Production Coordination

-Improves preventive maintenance scheduling

-Helps reduce defect rates and product waste

-Enables data-driven shift planning and training

-Supports Total Productive Maintenance (TPM) initiatives

8. 👥 Crew Supervision Simulation

```
# The previous code here caused a ValueError because the list length did not match the DataFrame length.
# Removing the erroneous lines to fix the error.
# df['CrewSize'] = [10, 9, 10, 11, 8]
# df['ProductivityPerWorker'] = df['ProductionVolume'] / df['CrewSize']
# print(df[['CrewSize', 'ProductivityPerWorker']])

# Add a 'CrewSize' column with appropriate data here if needed for further analysis.
# For example, you could generate random crew sizes for each shift:
import numpy as np
df['CrewSize'] = np.random.randint(8, 12, size=len(df)) # Example: random crew size between 8 and 11
df['ProductivityPerWorker'] = df['ProductionVolume'] / df['CrewSize']
print(df[['CrewSize', 'ProductivityPerWorker']].head())
```

```
   CrewSize  ProductivityPerWorker
0        11              18.363636
1         9              59.444444
2        10              96.000000
3        11              33.636364
4         9              22.888889
```

👷 Crew Size vs Productivity – Key Insights

📈 Productivity Varies Widely

-Even with similar Crew Sizes (mostly 9–11), Productivity Per Worker ranges significantly from ~18 to ~96 units, indicating uneven performance.

🔍 Not a Linear Relationship

-Larger crew size does not guarantee higher productivity. In fact, the most productive shift had 10 workers, not the largest team.

⚠️ Efficiency Gaps Exist

-Low productivity in some crews may signal underutilization, lack of coordination, or training needs.

🧠 Opportunity for Optimization

-By analyzing shifts with high per-worker output, best practices can be replicated across teams to boost efficiency.

💡 Operational Importance

-Supports balanced crew scheduling

-Identifies training or supervision needs

-Aids in labor cost vs output optimization

-Informs shift planning and crew talk topics

9. 📄 SOP Update Triggers

```
df['SOP_Update_Required'] = (df['DefectRate'] > 0.02) | (df['SafetyIncidents'] > 1)
print(df[['DefectRate', 'SafetyIncidents', 'SOP_Update_Required']])
```

```
      DefectRate  SafetyIncidents  SOP_Update_Required
0       3.121492                0                 True
1       0.819531                7                 True
2       4.514504                2                 True
3       0.638524                8                 True
4       3.867784                7                 True
...          ...              ...                  ...
3235    2.667570                3                 True
3236    0.751272                8                 True
3237    4.899756                5                 True
3238    4.057665                6                 True
3239    2.705502                4                 True

[3240 rows x 3 columns]
```

🛠️ Defect Rate vs Safety Incidents vs SOP Updates – Key Takeaways

⚠️ High Safety Incidents Correlate with High Defect Rates

-Several shifts show simultaneously high defect rates (≥4%) and multiple safety incidents (5–8) — suggesting process failures or training gaps.

📄 SOP Update Triggered

-In all these records, SOP_Update_Required = True, indicating the system flags such patterns for review and corrective action.

🔄 Strong Justification for Continuous Improvement

-These patterns underscore the need for up-to-date Standard Operating Procedures (SOPs) to reduce both product defects and safety risks.

🔍 Operational Importance

-Promotes proactive risk mitigation

-Supports CAPA (Corrective and Preventive Action) cycles

-Ensures audit readiness through SOP compliance

-Aligns with Total Productive Maintenance (TPM) and GMP (Good Manufacturing Practice) initiatives

Heat Map

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Sample Data
df = pd.DataFrame({
    'DefectRate': [3.12, 0.81, 4.51, 0.63, 3.86, 2.66, 0.75, 4.89, 4.05],
    'SafetyIncidents': [0, 7, 2, 8, 7, 3, 8, 5, 6],
    'SOP_Update_Required': [True]*9
})
```

```
# Group and pivot
heatmap_data = df.groupby('SafetyIncidents')['DefectRate'].mean().reset_index()
heatmap_pivot = heatmap_data.pivot_table(index='SafetyIncidents', values='DefectRate')

# Plot heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(heatmap_pivot, annot=True, cmap='Oranges', fmt=".2f")
plt.title("Avg Defect Rate by Safety Incident Level")
plt.ylabel("Safety Incidents")
plt.xlabel("Avg Defect Rate")
plt.tight_layout()
plt.show()
```

Explanation:

🔶 Heatmap Summary: Average Defect Rate by Safety Incident Level

-Color Scale: Darker orange indicates higher defect rates.

-Trend: Shifts with 5–6 safety incidents tend to have the highest average defect rates (>4.0).

- Low Incidents (e.g., 0 or 8) show lower defect rates, particularly at 8 incidents, which is counterintuitive and might require further investigation (e.g., reporting errors or shift differences).

📌 Importance:

- Identifies correlation between safety issues and product quality.
- Helps target SOP updates or training for shifts with high incidents and defects.
- Aids in prioritizing risk-based quality monitoring (RBQM) in pharma or manufacturing.
- Can support CAPA (Corrective and Preventive Action) planning by pinpointing problematic safety-performance zones.

10. ⚙️ TPM / Autonomous Maintenance Trigger

```
df['TPM_Flag'] = (df['MaintenanceHours'] > 5) | (df['DowntimePercentage'] > 1.5)
print(df[['MaintenanceHours', 'DowntimePercentage', 'TPM_Flag']])
```

```
      MaintenanceHours  DowntimePercentage  TPM_Flag
0                    9            0.052343      True
1                   20            4.908328      True
2                    1            2.464923      True
3                    8            4.692476      True
4                    9            2.746726      True
...                ...                 ...       ...
3235                16            0.987719      True
3236                11            0.178163      True
3237                 0            4.873429      True
3238                 6            0.071663      True
3239                13            4.803394      True

[3240 rows x 3 columns]
```

📊 Key Observations:

🛠️ Wide range of Maintenance Hours (0 to 20), with some low-maintenance shifts still showing high Downtime (>4.8%).

⚠️ Ineffective Maintenance Alerts: Downtime remains high even when TPM_Flag = True, indicating preventive tasks may not align with root causes.

✅ Low Downtime with Moderate Maintenance (e.g., 6–11 hours) suggests efficient TPM execution in certain shifts.

📌 Importance for Night Production Coordinator Role:

-Helps evaluate whether maintenance effort is yielding expected uptime.

-Supports adjustment of TPM scheduling to reduce over- or under-maintenance.

-Guides data-driven decision-making on SOP updates and shift planning.

-Essential for improving OEE (Overall Equipment Effectiveness) and productivity during night shifts

11. 📤 Outlook/PowerPoint-Style Summary Report

```
summary = df.describe()
summary.to_csv("/content/Shift_Performance_Summary.csv")
```

👹 MACHINE LEARNING PREDICTION & OUTCOMES

- Classification Objective: Predict DefectStatus (1 = Defective) using process and crew metrics.
- Key Predictors Identified:
  - DowntimePercentage, MaintenanceHours, SickCall, SafetyIncidents
- Business Outcomes:
  - Proactive SOP review triggers

- Optimized maintenance schedules (TPM)
- Shift performance benchmarking
- Reduced safety incidents and defect-related waste

🎞️ Trailer Documentation (Key Deliverables)

📊 Excel: Shift_Summary_Report.xlsx

📋 CSV: Shift_Performance_Summary.csv

📸 Visuals:

- Production vs Downtime Dual-Axis Chart
- Heatmap: Defect Rate by Safety Incidents
- Pareto: Maintenance vs Defect Rate

📝 SIC Notes Generator (Auto shift logs for meetings)

🛠️ SOP/TPM Flags – Triggers for CAPA and Lean reviews

🌐 Conceptual Enhancement – AGI (Artificial General Intelligence)

To simulate human-like coordination in manufacturing, the next phase could:

- Integrate AGI-inspired Agents that recommend actions using past shift logs and real-time production feeds.
- Build Contextual SOP Recommenders that learn from shift patterns.
  - Use Reinforcement Learning for adaptive crew and maintenance scheduling based on feedback loops.

📚 Reference

- Dataset: manufacturing_defect_dataset.csv (Google Drive)