MSP Analysis Report

Query:	Are we doing good ?
Generated:	2025-10-30 04:50:49
Complexity:	COMPLEX
Agents Used:	company_specific_ticket_agent, msp_insights_agent, license_audit_agent, financial_agent

Analysis Results

Here's a strategic analysis of your MSP company's performance:

Key Findings:

* **Service Performance is Moderate:** The MSP delivers consistent service with an average employee satisfaction score of 3.40/5 and an efficient average ticket resolution time of 5.60 hours. A total of 287 tickets were resolved.
* **Significant Cost Savings Opportunity:** There's a clear opportunity to reduce operational expenses by optimizing software licenses. 37 unused software licenses have been identified, many never used.
* **Compliance & Security Risks Identified:** 3 instances of employees accessing software outside their typical role permissions suggest potential compliance breaches or security vulnerabilities.
* **Critical Financial Data Gaps:** Key financial metrics such as overall MSP profitability, total annual revenue from active contracts, detailed payment trends (overdue, delayed, upcoming), and price revision impacts are currently unavailable due to errors in the financial reporting systems.

Financial Impact:

* **Potential Cost Savings:** A substantial **$11,113.88** can be saved immediately by re-evaluating or de-provisioning the identified 37 unused software licenses.
* **Unquantified Risk:** The 3 anomalous software access instances, while currently linked to free software, represent an unquantified risk for future compliance fines or security breaches if similar patterns occur with paid or sensitive applications.
* **Incomplete Financial Picture:** The errors in financial reporting prevent a comprehensive assessment of the company's overall financial health, profitability, cash flow, and revenue realization, making it difficult to fully answer "Are we doing good?" from a financial perspective.

Recommendations:

1. **Resolve Financial Reporting Errors (High Priority):**
* Immediately investigate and fix the `pandas query` errors in the `Msp Insights Agent` and `Financial Agent`. This is crucial for gaining visibility into overall profitability, revenue, and payment performance to make informed strategic decisions.
2. **Optimize Software Licensing (High Priority):**
* De-provision the 37 identified unused software licenses to realize the **$11,113.88** in cost savings.
* Implement a regular license usage review process to prevent future over-provisioning and ensure optimal resource allocation.
3. **Enhance Security and Compliance (Medium Priority):**

* Investigate the 3 instances of anomalous software access to understand their root cause and implement stricter access controls. This will mitigate potential compliance and security risks, especially for critical or paid software.
4. **Monitor and Improve Service Quality (Medium Priority):**
* While moderate, aim to improve the average employee satisfaction score from 3.40/5 by gathering more specific feedback and addressing any recurring issues.

Detailed Agent Data

Company Specific Ticket Agent

Here is a summary of the aggregated service performance metrics for all MSP client companies: **Overall Service Quality Assessment:** The aggregated data across all MSP client companies indicates a consistent level of service delivery. The average employee satisfaction score stands at approximately 3.40 out of 5, suggesting a moderate level of satisfaction. The average resolution time for tickets is approximately 5.60 hours, demonstrating efficiency in addressing issues. A total of 287 tickets have been resolved, reflecting the overall volume of support provided. **Detailed Metrics:** * **Average Employee Satisfaction:** 3.3970000000000007 * **Average Resolution Time (Hours):** 5.603 * **Total Resolved Tickets:** 287

Msp Insights Agent

■■ Error executing pandas query: columns overlap but no suffix specified: Index(['name'], dtype='object') Code: import pandas as pd import json # --- 1. Calculate Overall Profitability of the MSP --- # Total Annual Revenue from Active Contracts total_active_revenue = company_contract[company_contract['contract_status'] == 'Active']['annual_revenue'].sum() # Total Ticket Resolution Cost (assuming $25 per ticket) total_tickets_across_msp = company_contract['total_tickets'].sum() total_ticket_resolution_cost = total_tickets_across_msp * 25 # Total Software License Cost for all employees across all companies df_employees_software = customer_company_employees.copy() # Parse JSON string in 'assigned_software', handling potential NaN or malformed JSON df_employees_software['assigned_software_parsed'] = df_employees_software['assigned_software'].apply(lambda x: json.loads(x) if pd.notna(x) and isinstance(x, str) else []) # Explode the list of software dictionaries into separate rows df_exploded_software = df_employees_software.explode('assigned_software_parsed') # Filter out rows where 'assigned_software_parsed' is not a dictionary or is empty df_exploded_software_valid = df_exploded_software[df_exploded_software['assigned_software_parsed'].apply(lambda x: isinstance(x, dict) and 'name' in x and 'license_type' in x)] total_software_license_cost = 0.0 if not df_exploded_software_valid.empty: df_normalized_software = pd.json_normalize(df_exploded_software_valid['assigned_software_parsed']) df_merged_software_details = df_exploded_software_valid.reset_index(drop=True).join(df_normalized_software) # Merge with software_inventory to get license costs software_costs_df = df_merged_software_details.merge(software_inventory[['name', 'license_type', 'license_cost_usd']], on=['name', 'license_type'], how='left') # Sum license costs, treating NaN (for software not found) as 0 total_software_license_cost = software_costs_df['license_cost_usd'].sum() if pd.isna(total_software_license_cost): total_software_license_cost = 0.0 # Calculate Overall MSP Profitability overall_msp_profitability = total_active_revenue - total_ticket_resolution_cost - total_software_license_cost # --- 2. Total Annual Revenue from All Active Client Contracts --- # This value is already calculated as 'total_active_revenue' above. # --- 3. Summary of Payment History to Identify General Financial Trends --- total_amount_due_all_payments = payments['amount_due'].sum() total_amount_paid_all_payments = payments['amount_paid'].sum() # Payment status counts payment_status_counts = payments['status'].value_counts().reset_index() payment_status_counts.columns = ['Payment_Status', 'Count'] # Monthly payment trends payment_trends_monthly = payments.groupby('invoice_month').agg(Monthly_Amount_Due=('amount_due', 'sum'), Monthly_Amount_Paid=('amount_paid', 'sum')).reset_index() payment_trends_monthly.rename(columns={'invoice_month': 'Month'}, inplace=True) # --- Combine all results into a single DataFrame --- # Section 1: Key Financial Metrics financial_metrics_df = pd.DataFrame({ 'Metric': ['Overall MSP Profitability (USD)', 'Total Annual Revenue from Active Contracts (USD)', 'Total Ticket Resolution Cost (USD)', 'Total Software License Cost (USD)', 'Total Amount Due Across All Payments (USD)', 'Total Amount Paid Across All Payments (USD)'], 'Value': [overall_msp_profitability, total_active_revenue, total_ticket_resolution_cost, total_software_license_cost, total_amount_due_all_payments, total_amount_paid_all_payments] }) financial_metrics_df['Section'] = 'Key Financial Metrics' financial_metrics_df['Month'] = pd.NA financial_metrics_df['Monthly_Amount_Due'] = pd.NA financial_metrics_df['Monthly_Amount_Paid'] = pd.NA # Section 2: Payment Status Summary payment_status_summary_df = payment_status_counts.copy() payment_status_summary_df['Metric'] = 'Payment Status: ' + payment_status_summary_df['Payment_Status'] payment_status_summary_df.rename(columns={'Count': 'Value'}, inplace=True) payment_status_summary_df = payment_status_summary_df[['Metric', 'Value']]

```
payment_status_summary_df['Section'] = 'Payment Status Summary'
payment_status_summary_df['Month'] = pd.NA
payment_status_summary_df['Monthly_Amount_Due'] = pd.NA
payment_status_summary_df['Monthly_Amount_Paid'] = pd.NA # Section 3: Monthly Payment
Trends monthly_payment_trends_df = payment_trends_monthly.copy()
monthly_payment_trends_df['Section'] = 'Monthly Payment Trends'
monthly_payment_trends_df['Metric'] = 'Monthly Payment Data'
monthly_payment_trends_df['Value'] = pd.NA # Value column not applicable for these rows
# Reorder columns for monthly trends to match the combined structure
monthly_payment_trends_df = monthly_payment_trends_df[['Section', 'Month', 'Metric',
'Value', 'Monthly_Amount_Due', 'Monthly_Amount_Paid']] # Concatenate all sections into
a single result DataFrame result = pd.concat([financial_metrics_df,
payment_status_summary_df, monthly_payment_trends_df], ignore_index=True) # Final
column reordering for consistent output result = result[['Section', 'Metric', 'Value',
'Month', 'Monthly_Amount_Due', 'Monthly_Amount_Paid']]
```

License Audit Agent

```
This report identifies significant opportunities for cost savings by optimizing
software license usage and highlights potential compliance and security risks related to
anomalous software access. **Key Insights:** * **Substantial Potential Cost Savings:**
A total of **$11,113.88** can be saved by re-evaluating or de-provisioning 37 unused
software licenses. Many of these licenses have never been used, indicating potential
over-provisioning. * **Compliance and Risk Assessment:** 3 instances of employees
accessing software outside their typical role permissions have been identified,
indicating potential compliance breaches or security vulnerabilities. While the current
identified anomalous accesses are for free software, this pattern could lead to
significant risks with paid or sensitive applications. --- ### Unused Software Licenses
(Potential Cost Savings: $11,113.88) The following licenses are identified as unused,
detailing their last usage, days since last use, associated cost, and the reason for
inactivity: * **Rohan Singh:** `MySQL Workbench` - Last used:
2025-04-05T09:12:00+00:00, 207 days since last use, Cost: $0.0, Reason: Not used in last
90 days * **Vikram Iyer:** `Docker (Free)` - Last used: NEVER_USED, Days since last use:
NEVER_USED, Cost: $0.0, Reason: Never used * **Sneha Patel:** `Slack (Free)` - Last
used: NEVER_USED, Days since last use: NEVER_USED, Cost: $0.0, Reason: Never used *
**Amit Verma:** `QuickBooks` - Last used: NEVER_USED, Days since last use: NEVER_USED,
Cost: $360.0, Reason: Never used * **Priya Reddy:** `Google Analytics` - Last used:
NEVER_USED, Days since last use: NEVER_USED, Cost: $0.0, Reason: Never used * **Priya
Reddy:** `Canva` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $180.0,
Reason: Never used * **Karan Gupta:** `GitHub` - Last used: NEVER_USED, Days since last
use: NEVER_USED, Cost: $48.0, Reason: Never used * **Karan Gupta:** `Jira` - Last used:
NEVER_USED, Days since last use: NEVER_USED, Cost: $93.0, Reason: Never used * **Karan
Gupta:** `Figma` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $144.0,
Reason: Never used * **Meera Desai:** `Confluence` - Last used: NEVER_USED, Days since
last use: NEVER_USED, Cost: $66.0, Reason: Never used * **Meera Desai:** `Miro` - Last
used: NEVER_USED, Days since last use: NEVER_USED, Cost: $192.0, Reason: Never used *
**Arjun Mehta:** `LinkedIn Sales Navigator` - Last used: NEVER_USED, Days since last
use: NEVER_USED, Cost: $950.0, Reason: Never used * **Arjun Mehta:** `Microsoft Excel` -
Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $150.0, Reason: Never used
* **Divya Rao:** `Tableau` - Last used: NEVER_USED, Days since last use: NEVER_USED,
Cost: $840.0, Reason: Never used * **Divya Rao:** `AWS Sagemaker` - Last used:
NEVER_USED, Days since last use: NEVER_USED, Cost: $0.0, Reason: Never used * **Sameer
Khan:** `Microsoft 365 Admin` - Last used: NEVER_USED, Days since last use: NEVER_USED,
Cost: $0.0, Reason: Never used * **Sameer Khan:** `Norton Antivirus` - Last used:
NEVER_USED, Days since last use: NEVER_USED, Cost: $50.0, Reason: Never used * **Fatima
Ansari:** `Adobe Photoshop` - Last used: NEVER_USED, Days since last use: NEVER_USED,
Cost: $251.88, Reason: Never used * **Fatima Ansari:** `Canva` - Last used: NEVER_USED,
Days since last use: NEVER_USED, Cost: $180.0, Reason: Never used * **Rahul Nair:**
`Kubernetes` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $0.0,
Reason: Never used * **Rahul Nair:** `Jenkins` - Last used: NEVER_USED, Days since last
use: NEVER_USED, Cost: $0.0, Reason: Never used * **Rahul Nair:** `Terraform` - Last
used: NEVER_USED, Days since last use: NEVER_USED, Cost: $240.0, Reason: Never used *
**Sunita Joshi:** `Microsoft Excel` - Last used: NEVER_USED, Days since last use:
NEVER_USED, Cost: $150.0, Reason: Never used * **Sunita Joshi:** `SAP` - Last used:
NEVER_USED, Days since last use: NEVER_USED, Cost: $2500.0, Reason: Never used *
**Naveen Kumar:** `Microsoft Excel` - Last used: NEVER_USED, Days since last use:
NEVER_USED, Cost: $150.0, Reason: Never used * **Naveen Kumar:** `PowerBI` - Last used:
NEVER_USED, Days since last use: NEVER_USED, Cost: $120.0, Reason: Never used * **Pooja
Chavan:** `BambooHR` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost:
$99.0, Reason: Never used * **Pooja Chavan:** `Slack` - Last used: NEVER_USED, Days
since last use: NEVER_USED, Cost: $87.0, Reason: Never used * **Harish Patil:**
`Grammarly` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $144.0,
Reason: Never used * **Harish Patil:** `WordPress` - Last used: NEVER_USED, Days since
last use: NEVER_USED, Cost: $300.0, Reason: Never used * **Lakshmi Menon:** `Jira` -
Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $93.0, Reason: Never used
* **Lakshmi Menon:** `Figma` - Last used: NEVER_USED, Days since last use: NEVER_USED,
```

Cost: $144.0, Reason: Never used * **Alok Tiwari:** `Jira` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $93.0, Reason: Never used * **Alok Tiwari:** `Postman` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $1188.0, Reason: Never used * **Alok Tiwari:** `TestRail` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $372.0, Reason: Never used * **Ritu Agarwal:** `Salesforce` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $1800.0, Reason: Never used * **Ritu Agarwal:** `Microsoft Teams` - Last used: NEVER_USED, Days since last use: NEVER_USED, Cost: $60.0, Reason: Never used --- ### Anomalous Software Access (Total License Cost: $0.0) The following employees are using software outside their typical role permissions, indicating potential compliance or security risks: * **Rohan Singh** (UI/UX Designer) - `MySQL Workbench` (Community), Cost: $0.0, Reason: Role not typically allowed to use this software * **Vikram Iyer** (Backend Developer) - `Docker (Free)` (Free), Cost: $0.0, Reason: Role not typically allowed to use this software * **Sneha Patel** (HR Manager) - `Slack (Free)` (Free), Cost: $0.0, Reason: Role not typically allowed to use this software

Financial Agent

■■ Error executing pandas query: "['base_inflation', 'ticket_volume_impact', 'endpoint_scale_impact', 'payment_delay_penalty', 'happiness_adjustment', 'contract_length_discount'] not in index" Code: import pandas as pd # Define the common columns for the final result DataFrame all_cols = ['Summary_Type', 'company_id', 'company_name', 'contact_person', 'contact_email', 'Value', 'Date_Info', 'Days_Info', 'base_inflation', 'ticket_volume_impact', 'endpoint_scale_impact', 'payment_delay_penalty', 'happiness_adjustment', 'contract_length_discount'] # 1. Process Overdue Payments overdue_summary = overdue_payments.copy() overdue_summary['Summary_Type'] = 'Overdue Payment' overdue_summary['Value'] = overdue_summary['amount_due'] overdue_summary['Date_Info'] = overdue_summary['due_date'] overdue_summary['Days_Info'] = overdue_summary['days_overdue'] # Initialize factor breakdown columns to None for overdue payments for col in ['base_inflation', 'ticket_volume_impact', 'endpoint_scale_impact', 'payment_delay_penalty', 'happiness_adjustment', 'contract_length_discount']: overdue_summary[col] = None overdue_summary = overdue_summary[all_cols] # 2. Process Total Count of Delayed Payments total_delayed_count = len(delayed_payments) delayed_count_df = pd.DataFrame([{ 'Summary_Type': 'Delayed Payments Count', 'Value': float(total_delayed_count) # Cast to float for consistency with other 'Value' columns }]) # Fill other columns with None for delayed payments count for col in all_cols: if col not in delayed_count_df.columns: delayed_count_df[col] = None delayed_count_df = delayed_count_df[all_cols] # 3. Process Upcoming Pending Invoices upcoming_summary = upcoming_due_dates.copy() upcoming_summary['Summary_Type'] = 'Upcoming Pending Invoice' upcoming_summary['Value'] = upcoming_summary['amount_due'] upcoming_summary['Date_Info'] = upcoming_summary['due_date'] upcoming_summary['Days_Info'] = upcoming_summary['days_until_due'] # Initialize factor breakdown columns to None for upcoming payments for col in ['base_inflation', 'ticket_volume_impact', 'endpoint_scale_impact', 'payment_delay_penalty', 'happiness_adjustment', 'contract_length_discount']: upcoming_summary[col] = None upcoming_summary = upcoming_summary[all_cols] # 4. Process Proposed Price Revisions with detailed factor breakdowns # Normalize the 'factor_breakdown' column expanded_factors = safe_json_normalize(price_revisions, 'factor_breakdown') # Select relevant columns from price_revisions and concatenate with expanded factors price_summary = price_revisions[['company_id', 'company_name', 'contact_person', 'contact_email', 'annual_cost_change']].copy() price_summary = pd.concat([price_summary, expanded_factors], axis=1) price_summary['Summary_Type'] = 'Price Revision' price_summary['Value'] = price_summary['annual_cost_change'] price_summary['Date_Info'] = None # Not applicable for price revisions price_summary['Days_Info'] = None # Not applicable for price revisions price_summary = price_summary[all_cols] # Concatenate all summary dataframes into a single result DataFrame result = pd.concat([overdue_summary, delayed_count_df, upcoming_summary, price_summary], ignore_index=True)