**Share**

****

**You said:**

can you recall the details of project hack challenge 1, that we discussed

I don’t have a direct recall of "Project Hack Challenge 1" from our previous discussions. Could you provide some details or context about what it involved? That way, I can better assist you in retrieving relevant information or insights.

**You said:**

Efficient project execution hinges on maintaining accurate and feasible schedules. However, unexpected changes in resource availability often disrupt timelines, leading to delays, overutilisation or underutilisation of resources. This challenge seeks to develop a tool that dynamically adjusts project activity durations based on real-time resource data, recalculates critical paths and highlights changes in schedule impact. The tool aims to support project managers in ensuring optimal resource allocation and maintaining schedule feasibility

**ChatGPT said:**

Your challenge is focused on developing a **dynamic scheduling tool** that integrates real-time resource availability to adjust project timelines, recalculates critical paths, and provides insights on schedule impacts. This solution will help project managers maintain **optimal resource allocation** while minimizing delays and inefficiencies.

Here are key components to consider for the tool:

**Core Functionalities:**

1. **Real-time Resource Integration**
   * Connect with resource tracking systems (e.g., HR databases, ERP tools) to fetch availability data.
   * Monitor changes due to absence, task completion rates, or unexpected demand.
2. **Dynamic Schedule Adjustment**
   * Automatically update task durations based on available workforce, equipment, and constraints.
   * Use predictive analytics to foresee potential bottlenecks.
3. **Critical Path Recalculation**
   * Identify changes in task dependencies and recalculate the project’s **critical path** dynamically.
   * Highlight at-risk activities due to resource shortages or overload.
4. **Impact Visualization**
   * Provide **Gantt chart updates** and **heat maps** for over/underutilized resources.
   * Alert project managers on deviations from the baseline schedule.
5. **Scenario Analysis & Recommendations**
   * Offer alternative scheduling scenarios based on different resource allocation strategies.
   * Use AI/ML to suggest optimal adjustments for workload balancing.

**Technology Stack Options:**

* **Data Sources:** API integration with workforce management, ERP, or project management tools (MS Project, Primavera, Jira, etc.).
* **Backend Processing:** Python (Pandas for data handling, NetworkX for critical path recalculations).
* **Frontend/UI:** React.js for interactive dashboards.
* **Visualization Tools:** D3.js, Plotly, or Power BI for reporting insights.
* **Cloud Deployment:** AWS Lambda, Google Cloud Functions, or Azure for real-time processing.

Would you like assistance in drafting a project proposal or developing a prototype?

**You said:**

here are some pain points, 1. Schedule management schedules often fail to reflect real-time changes in resource availability. 2. Resource inefficiency, over or underutilising resources leads to missed deadlines or idle time. 3. Manual adjustments, current systems require extensive manual intervention for rescheduling, increasing risk of errors. 4. Limited Impact visibility, Lack of tools to compare original schedules with updated ones hinders effective decision-making.

**ChatGPT said:**

Your challenge addresses fundamental project scheduling inefficiencies. Here’s how we can align the **solution’s features** with the pain points you’ve identified:

**Pain Point Analysis & Solution Approach**

| **Pain Point** | **Solution Features** |
| --- | --- |
| **1. Schedule Management** – Schedules don’t reflect real-time resource availability. | **Real-time Integration:** The tool pulls live resource data (availability, workload) and automatically updates schedules accordingly. |
| **2. Resource Inefficiency** – Over/underutilization leads to delays or idle time. | **Automated Load Balancing:** The system reallocates resources dynamically to maintain even workloads and avoid bottlenecks. |
| **3. Manual Adjustments** – High risk of errors due to manual rescheduling. | **AI-Driven Optimization:** Uses machine learning to suggest adjustments, reducing human error and effort. |
| **4. Limited Impact Visibility** – No way to compare original vs. updated schedules. | **Impact Analysis & Visualization:** Provides side-by-side comparisons of baseline vs. adjusted schedules with key metrics. |

**How the Tool Works**

1. **Connects to Resource & Project Data Sources** – APIs fetch workforce availability, workload, and progress updates.
2. **Predicts & Adjusts Schedules** – AI-driven analysis recalculates durations and dependencies in real-time.
3. **Reoptimizes the Critical Path** – Ensures changes don’t compromise project deadlines.
4. **Visualizes Changes** – Interactive dashboards highlight schedule deviations and impact on project KPIs.

Would you like a **concept mockup** or a **technical roadmap** for development?

**You said:**

As a project manager, I want to quickly adapt schedules in response to real-time changes in resource availability, so that I can maintain project milestones and minimise disruptions. As a resource manager, I want to optimise resource allocation across multiple projects, so that I can prevent overallocation, avoid inefficiencies and ensure resource balance. As a stakeholder, i want clear and timely visibility into schedule adjustments, so that I can approve or revise resource distribution strategies with confidence.

**ChatGPT said:**

These user stories define the key functionalities of the **dynamic scheduling tool**. Here's how we can structure the solution:

**User Stories & Solution Features**

**1. Project Manager**

**Need:** Quickly adapt schedules in response to real-time resource availability.  
**Solution Features:**  
✅ **Automated Schedule Updates** – Real-time integration with resource data adjusts timelines dynamically.  
✅ **Critical Path Recalculation** – Ensures milestones are met by identifying dependencies at risk.  
✅ **Scenario Planning** – Provides alternative schedules based on different resource allocations.

**2. Resource Manager**

**Need:** Optimize resource allocation across multiple projects to prevent overallocation and inefficiencies.  
**Solution Features:**  
✅ **Resource Load Balancing** – AI-driven reallocation prevents bottlenecks and idle time.  
✅ **Multi-Project Dashboard** – Visibility into resource utilization across different projects.  
✅ **Forecasting & Alerts** – Identifies risks of shortages or excess capacity ahead of time.

**3. Stakeholder (Executive, Sponsor, PMO, etc.)**

**Need:** Clear and timely visibility into schedule adjustments to approve/revise resource strategies confidently.  
**Solution Features:**  
✅ **Baseline vs. Updated Schedule Views** – Easy comparison of original vs. adjusted schedules.  
✅ **Impact Analysis Reports** – Highlights key changes, risks, and recommendations.  
✅ **Approval Workflow** – Stakeholders can review and approve adjustments before implementation.

**Next Steps**

Would you like a **detailed feature roadmap**, **wireframe design**, or **prototype development plan**?

**You said:**

Business Context Project timelines can quickly become unfeasible in an environment of fluctuating resource availability. The ability to dynamically adapt schedules ensures that projects stay on track and resources are efficiently utilised. Providing stakeholders with clear insights into schedule adjustments builds confidence and supports timely corrective actions. Dataset Description Participants will be provided with a collection of spreadsheets detailing various resource availability scenarios to simulate real-world fluctuations. Additionally, an XER file from Primavera P6 will be included, representing the original project schedule, to serve as a baseline for comparisons and adjustments. Success Criteria 1.Dynamic Scheduling: Automatically adjust task durations in response to real-time resource data. 2.Impact Visualisation: Display side-by-side comparisons of original and updated schedules. 3.Critical Path Insights: Clearly show the recalculated critical paths post-adjustment. 4.Scalability: The tool must handle extensive, complex schedules seamlessly. 5.Ease of Use: Ensure the tool is intuitive and requires minimal training. Stretch Criteria: 1.Predictive Analytics: Incorporate trends to forecast future resource needs. 2.Optimization: Suggest optimisations to the schedules based on constraints and business priorities. 3.Integration: Theorise how to integrate with other project management and reporting tools.

**ChatGPT said:**

This challenge requires a **dynamic scheduling tool** that integrates real-time resource fluctuations, recalculates schedules, and provides actionable insights. Below is a structured approach based on your **business context, dataset description, and success criteria**.

**Proposed Solution Approach**

**1. Core Functionalities (Aligned with Success Criteria)**

✅ **Dynamic Scheduling**

* Parse Primavera P6 XER file to extract baseline schedules.
* Load resource availability data from spreadsheets in real-time.
* Automatically adjust task durations when resources fluctuate.

✅ **Impact Visualization**

* Provide side-by-side comparisons of **original vs. updated schedules**.
* Highlight task delays, shifts in dependencies, and changes in resource allocation.

✅ **Critical Path Insights**

* Recalculate the **Critical Path Method (CPM)** dynamically.
* Identify and flag **tasks at risk** due to resource constraints.

✅ **Scalability**

* Handle large-scale schedules with **hundreds of tasks and dependencies**.
* Use **optimized database storage** (SQL or NoSQL) for performance efficiency.

✅ **Ease of Use**

* User-friendly **web-based dashboard** or **Excel-based macro tool** for quick adoption.
* Interactive Gantt charts for clear visualization.

**2. Stretch Goals (Advanced Features)**

🚀 **Predictive Analytics**

* Analyze historical resource trends to **forecast potential bottlenecks**.
* Generate **early warnings** for anticipated schedule risks.

🚀 **Optimization Algorithms**

* Use AI/ML techniques (e.g., Genetic Algorithms, Linear Programming) to recommend **optimal resource allocations**.
* Prioritize business goals (cost, time, resource efficiency).

🚀 **Integration Possibilities**

* APIs to connect with **Primavera P6, MS Project, Jira, or SAP**.
* **Power BI / Tableau connectors** for real-time reporting.

**Technology Stack Recommendations**

🔹 **Backend:** Python (Pandas for data handling, NetworkX for critical path analysis)  
🔹 **Frontend:** React.js for interactive dashboards, Plotly for visualization  
🔹 **Database:** PostgreSQL for schedule tracking  
🔹 **Integration:** REST APIs for Primavera P6, Excel connectors

**Next Steps**

Would you like a **detailed project roadmap**, a **prototype design**, or help with **dataset processing**? 🚀

**You said:**

Benefit Enhanced decision-making is achieved by providing immediate visibility into schedule changes, allowing teams to make rapid, informed choices that minimise delays and ensure timely interventions. The tool also increases resource efficiency by aligning schedules with real-time availability, reducing downtime, preventing overloads, and optimising resource utilisation across projects. Automating rescheduling processes saves significant time, enabling teams to focus on higher-value activities and improving overall productivity. Additionally, transparent and data- driven scheduling fosters trust among stakeholders, promoting confidence in project timelines and deliverables.Page

**ChatGPT said:**

Your **dynamic scheduling tool** delivers significant benefits across multiple project management areas. Here’s how it aligns with key project objectives:

**🔹 Key Benefits & Impact**

✅ **1. Enhanced Decision-Making**

* **Immediate visibility** into schedule changes enables **faster, data-driven decisions**.
* Reduces the risk of **last-minute firefighting** by providing proactive alerts on scheduling conflicts.

✅ **2. Optimized Resource Utilization**

* Aligns schedules with **real-time resource availability**, **reducing downtime** and **preventing overloads**.
* **Cross-project resource balancing** ensures that no single project suffers due to allocation issues.

✅ **3. Increased Efficiency & Productivity**

* **Automates scheduling adjustments**, eliminating the need for **manual rescheduling**.
* Saves **hours of administrative work**, allowing project teams to focus on **higher-value strategic activities**.

✅ **4. Transparent & Trustworthy Scheduling**

* **Clear comparisons** of baseline vs. updated schedules foster **stakeholder confidence**.
* **Data-driven insights** provide a **rational basis for approvals**, reducing friction in project planning discussions.

**Next Steps**

Would you like to:  
🔹 Draft a **business case document** for stakeholder buy-in?  
🔹 Develop a **functional prototype**?  
🔹 Define a **detailed technical architecture**?

Let me know how you'd like to proceed! 🚀

**You said:**

can you define a technical architecture for this problem statement, utilising P6 data out, with SQL and looking to incorporate resource data on multiple Excel files, which will all be presented on a Power Bi dynamic dashboard being able to select from multiple projects if available and show the users the impact of the resource availability.

**ChatGPT said:**

Here’s a **technical architecture** for your **Dynamic Scheduling & Resource Impact Visualization Tool** using **Primavera P6, SQL, Excel resource data, and Power BI**.

**🔹 Technical Architecture Overview**

**Goal:** Create a **scalable** and **automated** system to:  
✔ Extract project schedules from **Primavera P6 (XER format)**  
✔ Integrate **resource data from multiple Excel files**  
✔ Store and process schedule changes in a **SQL database**  
✔ Dynamically visualize impacts via **Power BI**

**🔹 System Architecture Components**

**1. Data Sources**

🔹 **Primavera P6 (XER File) → SQL**

* Extract project schedules (Tasks, Durations, Dependencies, Critical Path).
* Convert into structured tables inside SQL for analysis.

🔹 **Excel Resource Files (Multiple Projects) → SQL**

* Load real-time resource availability updates.
* Ensure mapping to correct projects and tasks.

**2. Data Processing Layer (Backend & Automation)**

🔹 **Data Pipeline (Python/ETL Tools):**

* **XER Parser (Primavera P6 Data Extraction)**
  + Convert XER files into SQL-compatible formats.
* **Excel Processor (Resource Data Extraction)**
  + Aggregate resource availability from multiple Excel files.
* **Data Transformation (SQL Queries & Stored Procedures)**
  + Merge project schedules with real-time resource availability.
  + **Recalculate Critical Path** based on adjusted durations.
  + Generate **baseline vs. updated schedule impact analysis**.

**3. Database Layer (SQL Server / PostgreSQL)**

🔹 **Database Schema:**

* **Projects Table** – Stores multiple project details.
* **Tasks Table** – Includes task names, durations, dependencies.
* **Resources Table** – Maps personnel/equipment availability.
* **Schedule Changes Table** – Logs real-time adjustments.
* **Critical Path Table** – Stores recalculated paths dynamically.

🔹 **Key Functions in SQL:**

* **JOIN Queries** to match tasks with updated resource availability.
* **Stored Procedures** to recalculate task durations & critical path.

**4. Power BI (Dynamic Dashboards & User Interaction)**

✔ **Project Selection Filter** – Choose from multiple projects.  
✔ **Baseline vs. Updated Schedule** – See how changes impact the timeline.  
✔ **Critical Path Insights** – View bottlenecks & at-risk tasks.  
✔ **Resource Utilization Heatmaps** – Identify over- or under-utilized resources.  
✔ **Trend Analysis** – Predict future resource constraints using historical data.

**🔹 Technology Stack Recommendation**

🔹 **Data Extraction & Processing:** Python (Pandas, SQLAlchemy, NetworkX for Critical Path)  
🔹 **Database:** PostgreSQL / SQL Server (For scalable schedule storage & querying)  
🔹 **BI & Reporting:** Power BI (For real-time impact visualization)  
🔹 **Automation:** Azure Data Factory / Python Scripts (For scheduled ETL processes)

**🔹 Next Steps**

Would you like a **detailed ERD (Entity-Relationship Diagram) for SQL** or a **proof-of-concept (PoC) script for extracting and processing Primavera P6 data**? 🚀

**You said:**

Jarvis, we have until 17:00 today and 15:00 tomorrow to get this done, so streamline the solution based on time, with 6 people working to achieve this outcome.

**ChatGPT said:**

Got it! Given the **tight timeline** and **six-person team**, we need a **streamlined, high-impact approach** that delivers a functional prototype within the available working hours.

**🔹 Streamlined Plan for Delivery**

| **Task** | **Owner(s)** | **Time Estimate** |
| --- | --- | --- |
| ✅ **Data Extraction:** Parse Primavera P6 (XER) into SQL | Dev 1, Dev 2 | 3 hrs (Today) |
| ✅ **Resource Data Integration:** Load Excel resource files into SQL | Dev 3 | 2 hrs (Today) |
| ✅ **Critical Path Calculation:** SQL queries & stored procedures | Dev 4 | 3 hrs (Today) |
| ✅ **Power BI Initial Setup:** Connect SQL & create baseline dashboard | Dev 5, Dev 6 | 2 hrs (Today) |
| ✅ **Dynamic Impact Visualization:** Show updated schedules in Power BI | Dev 5 | 3 hrs (Tomorrow) |
| ✅ **Final Testing & Adjustments** | All | 2 hrs (Tomorrow) |

**🔹 Technical Execution Strategy**

✅ **Step 1: Extract Primavera P6 (XER) to SQL** *(3 hours - Today)*

* Use **XER parser script** (Python or SQL-based import)
* Store key tables: **Projects, Tasks, Dependencies**

✅ **Step 2: Load Resource Availability Data (Excel) into SQL** *(2 hours - Today)*

* Create **resource mapping table**
* Load into SQL with **ETL script (Python or Power Automate)**

✅ **Step 3: Recalculate Schedules Based on Resource Data** *(3 hours - Today)*

* SQL stored procedure **adjusts task durations** dynamically
* **Recalculate critical path** using **NetworkX (Python) or SQL queries**

✅ **Step 4: Build Power BI Dashboard (Baseline)** *(2 hours - Today)*

* Connect **SQL Database → Power BI**
* Display **project selection dropdown, baseline schedule**

✅ **Step 5: Implement Dynamic Impact Visualization** *(3 hours - Tomorrow)*

* Update Power BI to **compare baseline vs. updated schedules**
* **Highlight critical path changes**

✅ **Step 6: Testing & Final Adjustments** *(2 hours - Tomorrow)*

* Validate **data accuracy & schedule updates**
* **Ensure Power BI interactivity** (filters, drilldowns)

**🔹 Deployment Notes**

✔ **Minimal custom development** → Focus on **SQL queries + Power BI automation**  
✔ **Parallel workstreams** → Devs work **simultaneously** on XER parsing, SQL, and Power BI  
✔ **Iterative testing** → Validate **at each step** to avoid last-minute fixes

**🚀 Final Deliverable by 15:00 Tomorrow**

✅ **SQL Database with integrated schedule & resource data**  
✅ **Power BI Dashboard with dynamic project selection & impact analysis**  
✅ **Critical path recalculations & change tracking**

**Next Step:** Do you want a **code template for XER parsing & SQL procedures** right now? 💡

**You said:**

we will get to that in a minute mate, hold on.

**ChatGPT said:**

Got it, mate! Let me know when you're ready to dive in. 🚀

**You said:**

how can we turn on the resource histogram in P6 resource view

**ChatGPT said:**

To turn on the **Resource Histogram** in **Primavera P6**, follow these steps:

**🔹 Method 1: Using the Resource Usage Profile (Histogram) in the Resource Tab**

1. **Open Primavera P6** and go to your project.
2. Click on **"Resources"** in the top menu.
3. Select **"Resource Usage Profile"** from the dropdown.
4. The **histogram** will appear at the bottom of the screen.
5. If it’s not visible, click **"View" → "Show on Bottom" → "Resource Usage Profile"**.
6. Use the **Display Options** to customize the histogram (e.g., resource units, costs, time periods).

**🔹 Method 2: Using Activity Layout for Resource Histogram**

1. **Go to the "Activities" View**.
2. Click on **"View" → "Show on Bottom" → "Resource Usage Profile"**.
3. The **Resource Histogram** will appear, showing usage over time.
4. To filter by a specific resource, click **"Display" → "Filters" → Select a Resource**.

**🔹 Method 3: Using Resource Assignments View**

1. Navigate to **"Resource Assignments"** from the **Resource** menu.
2. Click **"View" → "Show on Bottom" → "Resource Usage Profile"**.
3. The histogram will display **resource loading over time**.

**💡 Pro Tip:**

* Adjust the **time scale** by right-clicking on the histogram and selecting **"Timescale"** to modify the view (e.g., daily, weekly, monthly).
* Click **"Customize Display"** to show units, costs, and other resource metrics.

Would you like help in **exporting this data to Power BI** for visualization? 🚀

**You said:**

change instructions please for Primavera P6 19.12

**ChatGPT said:**

For **Primavera P6 19.12**, follow these steps to turn on the **Resource Histogram** in the **Resource Usage Profile** view:

**🔹 Method 1: Enable Resource Histogram in Activities View**

1. **Open Primavera P6 19.12** and go to the **Activities** screen.
2. Click **"View"** in the top menu.
3. Select **"Show on Bottom Layout" → "Resource Usage Profile"**.
4. The **Resource Histogram** will appear at the bottom of the screen.
5. Right-click inside the histogram area and choose **"Resource Usage Profile Options"** to customize the display (e.g., units, costs, and timescale).

**🔹 Method 2: Enable Resource Histogram in Resource Assignments View**

1. Click on **"View"** in the top menu.
2. Select **"Resource Assignments"** from the dropdown.
3. Click **"View" → "Show on Bottom Layout" → "Resource Usage Profile"**.
4. The **Resource Histogram** will now appear at the bottom, displaying resource allocation over time.
5. Use the **Timescale** settings (right-click → "Timescale") to adjust the view (daily, weekly, monthly).

**🔹 Method 3: Enable Histogram in Resource Analysis View**

1. Open the **Resources** tab from the menu.
2. Click **"Resource Usage Profile"**.
3. If the histogram is not visible, go to **View → Show on Bottom Layout → Resource Usage Profile**.
4. Use the **Display Options** to adjust what is shown (e.g., by resource, project, or role).

**💡 Pro Tips:**

✔ **To filter specific resources**, click **"Display" → "Filters"** and select the required resource(s).  
✔ **To export the histogram**, click **"Print Preview"** or take a **screenshot** for reporting.

Would you like help **extracting this data into SQL or Power BI**? 🚀

**You said:**

right we need to start, thinking about how and what data to we put into SQL, to correlate with the XLS data, which is currently set as AP10-2012, which is Jan 2013. what columns would I need to pull out of P6 into SQL to make this solution work, in terms of having a dynamic dashboard selecting scenarios affecting the resource assignments, which will impact the critical path and end date of the project.

**ChatGPT said:**

**🔹 SQL Data Structure for Dynamic Resource Impact Analysis**

To **correlate Primavera P6 (XER) data with Excel resource availability (AP10-2012 format)**, we need to extract key **schedule and resource-related** data into a structured SQL database.

**1️⃣ Key Data to Extract from P6 into SQL**

| **Table Name** | **Required Columns** | **Purpose** |
| --- | --- | --- |
| **Projects** | ProjectID, ProjectName, StartDate, EndDate | Identifies multiple projects for selection in Power BI |
| **Activities (Tasks)** | ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID | Stores tasks, durations, and identifies critical path activities |
| **Relationships (Dependencies)** | PredID, SuccID, Lag, RelationshipType | Defines dependencies for critical path calculations |
| **Resources** | ResourceID, ResourceName, Role, MaxUnits, CalendarID | Defines available resources and their allocation limits |
| **Resource Assignments** | ActivityID, ResourceID, ProjectID, UnitsPerDay, TotalUnits, BudgetedUnits | Links resources to activities and defines workloads |
| **Calendars** | CalendarID, CalendarName, WorkWeekHours, Holidays | Ensures correct time-based calculations |
| **Scenario Data (Excel Import)** | ScenarioID, ResourceID, AvailableUnits, WeekStart, WeekEnd | Holds real-time resource changes per scenario |

**2️⃣ SQL Schema & Relationships**

🔹 Projects**↔**Activities (One-to-Many)  
🔹 Activities**↔**Relationships (One-to-Many)  
🔹 Activities**↔**Resource Assignments**↔**Resources (Many-to-Many)  
🔹 Calendars**link to**Activities**and**Resources (Ensures time adjustments)  
🔹 Scenario Data**joins**Resource Assignments**to simulate real-time changes**

**3️⃣ Dynamic Dashboard Integration (Power BI)**

✔ **Project Selection Dropdown** → Filters SQL tables for project-specific insights  
✔ **Scenario Comparison (Baseline vs. Adjusted)** → Shows resource impact per scenario  
✔ **Critical Path Changes** → Recalculate after resource adjustments  
✔ **End Date Prediction** → Shows delays or improvements in project completion

**🚀 Next Steps**

Would you like:  
🔹 **SQL DDL script** to create tables?  
🔹 **Python ETL script** to extract XER data into SQL?  
🔹 **Power BI mockup for interactive dashboard?**

Let me know how you want to proceed! 🚀

**You said:**

do we need to create the tables in SQL first, before we extract the XER data from P6?

**ChatGPT said:**

Yes, we **must** create the tables in SQL **before extracting the XER data** from Primavera P6.

**🔹 Why Create SQL Tables First?**

✅ **Ensures Data Integrity** → Prevents missing relationships when loading data.  
✅ **Defines Data Structure** → Avoids inconsistencies in column formats (dates, numbers, etc.).  
✅ **Optimizes ETL Process** → Extracting XER into a predefined schema makes data mapping easier.  
✅ **Speeds Up Power BI Integration** → Structured SQL tables allow for efficient queries and filtering.

**🔹 Step-by-Step Approach**

**1️⃣ Create SQL Tables**

* Define **Projects, Activities, Relationships, Resources, Resource Assignments, Calendars, and Scenario Data** in SQL.
* Ensure **Primary & Foreign Keys** for proper relational mapping.

**2️⃣ Extract & Transform XER Data**

* Use a **Python script or SQL Import Tool** to parse XER and insert data into SQL.

**3️⃣ Load Excel Resource Scenario Data**

* Import real-time **resource availability from Excel** into SQL’s **Scenario Data Table**.

**4️⃣ Query & Analyze in Power BI**

* Generate reports comparing **Baseline vs. Adjusted Schedule**.

**🚀 Next Steps**

Would you like a **SQL script (DDL) to create tables now**, or do you want to **define exact column types first**?

**You said:**

we need a step by step walk through to connect the SQL environment to VS Code.

**ChatGPT said:**

**🔹 Step-by-Step Guide to Connect SQL Environment to VS Code**

To set up **SQL Server (PostgreSQL or MySQL) with VS Code**, follow these steps:

**1️⃣ Install Required Tools**

🔹 **For Microsoft SQL Server (MSSQL):**

1. **Download & Install SQL Server**
   * Install **SQL Server Express** or **SQL Server Developer Edition** [from here](https://www.microsoft.com/en-us/sql-server/sql-server-downloads).
   * Install **SQL Server Management Studio (SSMS)** (optional for easier management).
2. **Install SQL Server Extension in VS Code**
   * Open VS Code
   * Go to **Extensions (**Ctrl + Shift + X**)**
   * Search for **SQL Server (mssql)**
   * Click **Install**
3. **Install SQL Server Command Line Tools** (SQLCMD)
   * Download from: [Microsoft Docs](https://docs.microsoft.com/en-us/sql/tools/sqlcmd-utility)

🔹 **For PostgreSQL:**

1. **Download & Install PostgreSQL**
   * Install **PostgreSQL** from [PostgreSQL Official Site](https://www.postgresql.org/download/).
   * Set up a **username, password, and default database** during installation.
2. **Install PostgreSQL Extension in VS Code**
   * Open VS Code
   * Go to **Extensions (**Ctrl + Shift + X**)**
   * Search for **PostgreSQL**
   * Click **Install**

🔹 **For MySQL:**

1. **Download & Install MySQL Server**
   * Install **MySQL Server** from [MySQL Official Site](https://dev.mysql.com/downloads/).
   * Use **MySQL Workbench** (optional) for GUI-based management.
2. **Install MySQL Extension in VS Code**
   * Open VS Code
   * Go to **Extensions (**Ctrl + Shift + X**)**
   * Search for **MySQL**
   * Click **Install**

**2️⃣ Connect VS Code to SQL Server**

🔹 **For MSSQL:**

1. Press Ctrl + Shift + P (Command Palette).
2. Search for **"MS SQL: Connect"** and select it.
3. Choose **"Create a new connection"**.
4. Enter **server name** (e.g., localhost,1433).
5. Choose **Authentication Type:**
   * Use **SQL Login**
   * Enter **Username & Password**
6. Select **Database** (e.g., master or your project DB).
7. Press **Enter** to connect.
8. Start writing SQL queries in VS Code.

🔹 **For PostgreSQL:**

1. Open VS Code.
2. Press Ctrl + Shift + P, search for **"PostgreSQL: Add New Connection"**.
3. Enter **server details** (e.g., localhost:5432).
4. Provide **Database Name, Username, Password**.
5. Click **Connect**.
6. Start executing queries in VS Code.

🔹 **For MySQL:**

1. Open VS Code.
2. Go to the **MySQL extension**.
3. Click **"Add New Connection"**.
4. Enter **Host** (e.g., localhost), **Port** (default 3306), **Username**, **Password**.
5. Choose **Database** and connect.
6. Start running queries in VS Code.

**3️⃣ Test Connection**

Run the following queries to check connectivity:

✅ **For MSSQL:**

sql

Copy

SELECT name FROM sys.databases;

✅ **For PostgreSQL:**

sql

Copy

SELECT datname FROM pg\_database;

✅ **For MySQL:**

sql

Copy

SHOW DATABASES;

**🚀 Next Steps**

Would you like **help creating the SQL database schema** or **guidance on writing queries in VS Code?**

**You said:**

ok, give me the instructions again slowly, for MSSQL

**ChatGPT said:**

**🔹 Step-by-Step Guide to Connect Microsoft SQL Server (MSSQL) to VS Code**

Follow these steps carefully to set up and connect **MSSQL** with **VS Code**.

**1️⃣ Install Microsoft SQL Server and Required Tools**

**Step 1: Download & Install SQL Server**

1. Go to the official Microsoft SQL Server download page:  
   👉 [SQL Server Download](https://www.microsoft.com/en-us/sql-server/sql-server-downloads)
2. Download **SQL Server Express** (lightweight) or **SQL Server Developer** (full version).
3. Run the installer and choose **Basic Installation**.
4. Follow the prompts and install SQL Server.
5. When finished, **note down** the **server name** shown in the installation summary (e.g., DESKTOP-XXXX\SQLEXPRESS).

**Step 2: Install SQL Server Management Studio (SSMS) (Optional but Recommended)**

1. Go to: 👉 SSMS Download
2. Download and install **SSMS** to manage databases more easily.

**Step 3: Install SQL Server Command Line Tools (SQLCMD) (Optional for CLI Users)**

1. Download SQLCMD tools from: 👉 [SQLCMD Installation Guide](https://docs.microsoft.com/en-us/sql/tools/sqlcmd-utility)
2. Install it and ensure it runs correctly by typing:

sh

Copy

sqlcmd -?

If it shows available commands, it's installed correctly.

**2️⃣ Install SQL Server Extension in VS Code**

**Step 4: Open VS Code and Install the MSSQL Extension**

1. Open **VS Code**.
2. Click on **Extensions** (or press Ctrl + Shift + X).
3. Search for **"SQL Server (mssql)"**.
4. Click **Install**.

**3️⃣ Connect VS Code to SQL Server**

**Step 5: Open VS Code Connection Window**

1. Press Ctrl + Shift + P to open the **Command Palette**.
2. Type: **"MS SQL: Connect"** and select it.
3. Choose **"Create a new connection"**.

**Step 6: Enter Server Details**

1. When prompted, enter your **SQL Server Name**:
   * If using **SQL Express**, enter:

Copy

localhost\SQLEXPRESS

* + If using **default SQL Server**, enter:

nginx

Copy

localhost

* + If using a **named instance**, enter:

Copy

DESKTOP-XXXX\YourInstanceName

1. Choose **"SQL Login"** as the Authentication Type.
2. Enter **Username** and **Password** (from SQL Server setup).
3. Select a **Database** (or press **Enter** to use the default master database).

**Step 7: Test the Connection**

1. Open a new **SQL file** in VS Code (File → New File → Select SQL).
2. Run the following query to test:

sql

Copy

SELECT name FROM sys.databases;

1. If it returns a list of databases, the connection is **successful**.

**4️⃣ Next Steps**

Now that you're connected, do you want:  
🔹 A **SQL script to create tables for XER data**?  
🔹 A **query to check and load resource data from Excel**?

Let me know how you'd like to proceed! 🚀

**You said:**

its saying I need to have winget installed first...do you agree?

**ChatGPT said:**

Yes, if you're using **Windows Package Manager (**winget**)** to install SQL Server or related tools, you may need to have winget installed first.

**🔹 Check if Winget is Installed**

1. Open **Command Prompt (CMD)** or **PowerShell**.
2. Type:

sh

Copy

winget --version

1. If it returns a version number, **winget is installed**.
2. If it says **"command not found"**, you need to install it.

**🔹 Install Winget (If Not Installed)**

**Method 1: Update Windows (Recommended)**

* winget is included in **Windows 10 (Build 2004+) and Windows 11**.
* Check for updates:
  1. Go to **Settings → Update & Security → Windows Update**.
  2. Click **Check for updates**.
  3. Install the latest updates, then restart your PC.

**Method 2: Manually Install Winget**

1. Download the latest **App Installer** from GitHub:  
   👉 [Windows Package Manager GitHub](https://github.com/microsoft/winget-cli/releases)
2. Download the .msixbundle file.
3. Double-click to install it.

**🔹 Verify Winget Installation**

* Run:

sh

Copy

winget --info

* If it works, proceed with installing SQL Server.

**🔹 Install SQL Server Using Winget**

Once winget is installed, you can install SQL Server with:

sh

Copy

winget install Microsoft.SQLServerExpress

Would you like **help setting up SQL Server manually** instead of using winget? 🚀

**You said:**

i already have SQL server 22 and SQMSS already installed on my machine, we are not doing that again Jarvis......too long last night bud.

**ChatGPT said:**

Got it, mate! No need to go through that pain again. Since **SQL Server 2022 and SSMS** are already installed, let's move straight to **connecting VS Code to SQL Server** and getting our tables set up.

**🔹 Connect VS Code to SQL Server (MSSQL 2022)**

Since you already have SQL Server installed, let's **skip installation** and go straight to setting up **VS Code connectivity**.

**1️⃣ Install SQL Server Extension in VS Code**

1. Open **VS Code**.
2. Click on **Extensions** (or press Ctrl + Shift + X).
3. Search for **"SQL Server (mssql)"**.
4. Click **Install**.

**2️⃣ Open a New SQL Connection in VS Code**

1. Press Ctrl + Shift + P (Command Palette).
2. Type: **"MS SQL: Connect"** and select it.
3. Choose **"Create a new connection"**.
4. Enter your **SQL Server Name** (one of the following):
   * Default Instance → localhost
   * SQL Express → localhost\SQLEXPRESS
   * Named Instance → DESKTOP-XXXX\YourInstanceName
5. Choose **Authentication Type**:
   * If using **Windows Authentication**, select **"Integrated"**.
   * If using **SQL Login**, enter **your SQL username & password**.
6. Select **Database** (use master for now).
7. Press **Enter** to connect.

**3️⃣ Test SQL Connection**

1. Open a **new file** in VS Code (File → New File → Select SQL).
2. Run the following test query:

sql

Copy

SELECT name FROM sys.databases;

1. Click **Run Query (**Ctrl + Shift + E**)**.
2. If it returns a list of databases, 🎉 **connection is successful**.

**🔹 Next Steps**

Now that you're connected, do you want:  
✅ A **SQL script to create tables for P6 XER data?**  
✅ A **query to load resource data from Excel?**  
Let me know the next move, mate! 🚀

**You said:**

i have done the connection, but its now saying, an sql editor must have focus before executing a command

**ChatGPT said:**

Great! Now that you've successfully connected VS Code to SQL Server, let's fix the **"SQL editor must have focus before executing a command"** issue.

**🔹 Fix: Ensure You're in an Active SQL File**

Before running a query, make sure you have an **active SQL editor window** in focus. Follow these steps:

**1️⃣ Open a New SQL Query File in VS Code**

1. Open **VS Code**.
2. Click **File → New File**.
3. Save it as **test.sql** (Ctrl + S).
4. In the **bottom-right corner**, ensure **SQL language mode is active** (if not, click it and select "SQL").

**2️⃣ Select the Correct SQL Connection in VS Code**

1. Click on the **database connection icon** at the bottom left of VS Code.
2. Select your **SQL Server instance** (e.g., localhost\SQLEXPRESS).
3. Ensure it shows **"Connected"** in the status bar.

**3️⃣ Run the SQL Query**

Now, try running this test query:

sql

Copy

SELECT name FROM sys.databases;

1. Click anywhere inside the SQL query window.
2. Press Ctrl + Shift + E to execute the command.

**🔹 Alternative Fix: Manually Switch to SQL Editor**

If the error persists:

1. Click inside the **SQL script window** to activate it.
2. Try running the query again using Ctrl + Shift + E.

Now, let me know if you're **ready to create tables for the XER data** or need further troubleshooting! 🚀

**You said:**

i havent put my database credentials in yet, to be able to make the connection

**ChatGPT said:**

**🔹 Add Your Database Credentials in VS Code (MSSQL Connection)**

Since you haven’t entered your **database credentials**, let’s go through the process step by step.

**1️⃣ Open VS Code and Start a New Connection**

1. **Press**Ctrl + Shift + P to open the **Command Palette**.
2. Type **"MS SQL: Connect"** and select it.
3. Choose **"Create a new connection"**.

**2️⃣ Enter SQL Server Connection Details**

* **Server Name:**
  + Default instance → localhost
  + SQL Express → localhost\SQLEXPRESS
  + Named instance → DESKTOP-XXXX\YourInstanceName
* **Authentication Type:**
  + If using **SQL Server Authentication**, select **"SQL Login"**.
  + If using **Windows Authentication**, select **"Integrated"** (No username/password needed).
* **Username:** Enter your **SQL Server login** (e.g., sa or your custom user).
* **Password:** Enter your **SQL Server password** (hidden for security).

**3️⃣ Select Database**

* If prompted, select **"master"** (or your specific project database).
* Press **Enter** to confirm.

**4️⃣ Save Connection for Reuse (Optional)**

If you don’t want to enter credentials every time:

1. Open **VS Code Settings** (Ctrl + ,).
2. Search for **"mssql connections"**.
3. Add a **saved connection profile** in settings.json.

Example JSON format:

json

Copy

{ "mssql.connections": [ { "server": "localhost", "database": "YourDatabaseName", "user": "YourUsername", "password": "YourPassword", "authenticationType": "SqlLogin" } ] }

**5️⃣ Test Connection with a Query**

Now, open a new **SQL file** and run:

sql

Copy

SELECT name FROM sys.databases;

* Press Ctrl + Shift + E to execute.

If it works, **you are successfully connected!** 🎉

**🔹 Next Steps**

Do you want to:  
✅ **Create the SQL tables for Primavera XER data?**  
✅ **Import the Excel resource data into SQL?**

Let me know what’s next! 🚀

**You said:**

its not allowing me to put my database credentials in...I just keep getting the you must have focus error.

**ChatGPT said:**

Alright, let's troubleshoot this step by step and make sure you can enter your **SQL Server credentials in VS Code** properly.

**🔹 Step 1: Manually Add Connection Using Command Palette**

1. **Press**Ctrl + Shift + P to open the **Command Palette**.
2. Type **"MS SQL: Connect"** and select it.
3. Choose **"Create a new connection"**.
4. When prompted for **Server Name**, enter:
   * Default: localhost
   * SQL Express: localhost\SQLEXPRESS
   * Named instance: DESKTOP-XXXX\YourInstanceName
5. For **Authentication Type**, select **"SQL Login"**.
6. Enter your **SQL Username** (e.g., sa or your SQL login).
7. Enter your **SQL Password** (hidden for security).
8. Select **Database** (or press **Enter** to use master).

If this fails and gives the **"Must have focus error"**, proceed to **Step 2**.

**🔹 Step 2: Open SQL File in VS Code (Ensure Focus on SQL Editor)**

If the issue persists, it's likely because **VS Code doesn’t recognize an active SQL file**.

1. Open VS Code.
2. Click **File → New File**.
3. Save it as **test.sql** (Ctrl + S).
4. Ensure **SQL language mode** is active:
   * Check the **bottom-right corner** of VS Code.
   * If it’s not set to **SQL**, click it and select **SQL**.

Now try **Step 1 again**.

**🔹 Step 3: Manually Configure Connection in Settings**

If VS Code **still won’t let you enter credentials**, try manually adding them:

1. Open **VS Code Settings** (Ctrl + ,).
2. Search for **"mssql connections"**.
3. Click **Edit in settings.json**.
4. Add this connection profile:

json

Copy

{ "mssql.connections": [ { "server": "localhost", "database": "master", "user": "YourUsername", "password": "YourPassword", "authenticationType": "SqlLogin" } ] }

1. Restart VS Code and test connection again.

**🔹 Step 4: Run a Direct SQLCMD Test (Check If Credentials Work Outside VS Code)**

If VS Code still doesn't connect, we need to **verify if your SQL Server login works manually**.

1. Open **Command Prompt (**cmd**)**.
2. Run this command to connect manually:

sh

Copy

sqlcmd -S localhost -U YourUsername -P YourPassword

1. If successful, try running:

sql

Copy

SELECT name FROM sys.databases; GO

1. If this works, the issue is **VS Code-specific**, and we can try **reinstalling the MSSQL extension**.

**🔹 Step 5: Reinstall SQL Server Extension in VS Code (Final Fix)**

If nothing works, let’s **reset the MSSQL extension** in VS Code.

1. Open **Extensions (**Ctrl + Shift + X**)**.
2. Search for **"SQL Server (mssql)"**.
3. Click **Uninstall**, then **Reinstall**.
4. Restart VS Code.
5. Try **Step 1 again**.

**🚀 Final Check**

Now, open a new **SQL file** and test with this query:

sql

Copy

SELECT name FROM sys.databases;

If you see database names, **you're fully connected!** 🎉

Let me know which step worked or if you need more help! 🔧🚀

**You said:**

Jarvis, Your not getting it.....I am pressing connect, but it just stops blank, and says, An SQL editor must have focus to execute a command, am I missing a module from VS COde?

**ChatGPT said:**

Got it, mate! If VS Code **stops blank** when you try to connect and shows **"An SQL editor must have focus to execute a command"**, you're likely missing a **critical module or configuration step**. Let's go through a **full fix approach** step by step.

**🔹 Step 1: Verify You Have the Right Extensions Installed**

Since you're using **MSSQL**, you need:

**✅ Ensure You Have the SQL Server Extension Installed**

1. Open **VS Code**.
2. Press Ctrl + Shift + X to open the **Extensions Marketplace**.
3. Search for **"SQL Server (mssql)"**.
4. If it’s installed, **Uninstall it** and then **Reinstall it** (to ensure no corruption).
5. Restart VS Code after reinstalling.

**🔹 Step 2: Ensure You Have the Required Node.js Module**

The **SQL Server extension relies on Node.js**. If it’s missing, the connection will fail silently.

**✅ Check if Node.js is Installed**

1. Open **Command Prompt (cmd)** or **PowerShell**.
2. Type:

sh

Copy

node -v

1. If it returns a version number (v16.x or higher), **Node.js is installed**.
2. If **not installed**, download and install it from 👉 [Node.js Official Site](https://nodejs.org/).

**🔹 Step 3: Open a New SQL File (Ensure Focus on SQL Editor)**

The **"must have focus" error** usually happens when VS Code doesn’t recognize an active SQL file.

**✅ Open a Proper SQL Editor in VS Code**

1. Open **VS Code**.
2. Click **File → New File**.
3. Save it as **test.sql** (Ctrl + S).
4. **Ensure the file is set to SQL Mode**:
   * Look at the **bottom-right corner** of VS Code.
   * If it’s not set to **SQL**, click it and select **SQL**.

**🔹 Step 4: Try a Manual Connection via JSON (Bypass UI Issue)**

If the connection UI **isn’t responding**, manually define the connection.

**✅ Manually Configure Connection in VS Code Settings**

1. Open **VS Code Settings** (Ctrl + ,).
2. Search for **"mssql connections"**.
3. Click **"Edit in settings.json"**.
4. Add this configuration:

json

Copy

{ "mssql.connections": [ { "server": "localhost", "database": "master", "user": "YourUsername", "password": "YourPassword", "authenticationType": "SqlLogin" } ] }

1. Restart VS Code and test connection again.

**🔹 Step 5: Manually Test SQL Connection via Terminal**

If VS Code still **doesn’t connect**, let’s **check if SQL Server is reachable**:

1. Open **Command Prompt (**cmd**)**.
2. Run this command:

sh

Copy

sqlcmd -S localhost -U YourUsername -P YourPassword

1. If you **connect successfully**, SQL Server is working fine, and the issue is **inside VS Code**.

**🔹 Step 6: Fully Reset VS Code SQL Environment (Final Fix)**

If all else fails, let's **fully reset the VS Code SQL extension**:

1. Open **VS Code**.
2. Go to **Extensions (**Ctrl + Shift + X**)**.
3. Uninstall **SQL Server (mssql)**.
4. Close **VS Code**.
5. Open **Command Prompt** and run:

sh

Copy

code --disable-extensions

1. Reinstall **SQL Server (mssql)** extension.
2. Restart VS Code and try connecting again.

**🚀 Final Check**

Now, open a **SQL file** and run this test query:

sql

Copy

SELECT name FROM sys.databases;

If this works, 🎉 **problem solved!**

Let me know **which step worked** or if it’s still giving issues! 🚀🔧

**You said:**

right bud, we are connected, now we need to create some tables....lets go...what do you suggest for the challenge 1 solution as discussed earlier

**ChatGPT said:**

Great, mate! Now that we're **connected**, let's design the **SQL tables** needed for the **Challenge 1 solution**.

**🔹 Database Schema Overview**

We'll create **six core tables** to store:  
✔ **Projects** (to manage multiple project schedules)  
✔ **Activities** (tasks from Primavera P6)  
✔ **Relationships** (dependencies between tasks)  
✔ **Resources** (workers, materials, etc.)  
✔ **Resource Assignments** (links resources to activities)  
✔ **Scenario Data** (Excel-imported resource availability for dynamic scheduling)

**🔹 1️⃣ Create**Projects**Table**

Stores general project information.

sql

Copy

CREATE TABLE Projects ( ProjectID INT PRIMARY KEY IDENTITY(1,1), ProjectName NVARCHAR(255) NOT NULL, StartDate DATE, EndDate DATE );

**🔹 2️⃣ Create**Activities**Table**

Stores tasks from Primavera P6.

sql

Copy

CREATE TABLE Activities ( ActivityID INT PRIMARY KEY IDENTITY(1,1), ProjectID INT FOREIGN KEY REFERENCES Projects(ProjectID), ActivityName NVARCHAR(255) NOT NULL, StartDate DATE, FinishDate DATE, Duration INT, CriticalFlag BIT DEFAULT 0, CalendarID INT );

**🔹 3️⃣ Create**Relationships**Table**

Stores dependencies between tasks (Critical Path Analysis).

sql

Copy

CREATE TABLE Relationships ( RelationshipID INT PRIMARY KEY IDENTITY(1,1), PredID INT, *-- Predecessor Activity* SuccID INT, *-- Successor Activity* Lag INT DEFAULT 0, RelationshipType NVARCHAR(50) CHECK (RelationshipType IN ('FS', 'SS', 'FF', 'SF')), *-- Finish-Start, Start-Start, etc.* FOREIGN KEY (PredID) REFERENCES Activities(ActivityID), FOREIGN KEY (SuccID) REFERENCES Activities(ActivityID) );

**🔹 4️⃣ Create**Resources**Table**

Defines available workforce/equipment.

sql

Copy

CREATE TABLE Resources ( ResourceID INT PRIMARY KEY IDENTITY(1,1), ResourceName NVARCHAR(255) NOT NULL, Role NVARCHAR(100), MaxUnits INT, *-- Maximum available units (e.g., hours per day)* CalendarID INT );

**🔹 5️⃣ Create**ResourceAssignments**Table**

Links resources to tasks and tracks workload.

sql

Copy

CREATE TABLE ResourceAssignments ( AssignmentID INT PRIMARY KEY IDENTITY(1,1), ActivityID INT FOREIGN KEY REFERENCES Activities(ActivityID), ResourceID INT FOREIGN KEY REFERENCES Resources(ResourceID), ProjectID INT FOREIGN KEY REFERENCES Projects(ProjectID), UnitsPerDay DECIMAL(5,2), TotalUnits DECIMAL(10,2), BudgetedUnits DECIMAL(10,2) );

**🔹 6️⃣ Create**ScenarioData**Table**

Stores real-time resource adjustments (from Excel).

sql

Copy

CREATE TABLE ScenarioData ( ScenarioID INT PRIMARY KEY IDENTITY(1,1), ResourceID INT FOREIGN KEY REFERENCES Resources(ResourceID), AvailableUnits DECIMAL(5,2), WeekStart DATE, WeekEnd DATE );

**🔹 Next Steps**

✔ Run these **SQL commands in VS Code** to create the tables.  
✔ Once done, we can **import Primavera P6 (XER) data** and **Excel resource scenarios**.

Let me know **once tables are set up**, and we’ll move to **data extraction!** 🚀

**You said:**

I have created all the tables as per your instruction, but I cant see them in the dbo.TABLES list....or are they somewhere else?

**ChatGPT said:**

Good job creating the tables! If you **can’t see them in**dbo.TABLES, let’s check where they are and fix the issue.

**🔹 Step 1: Verify Tables Exist in the Correct Database**

By default, SQL Server creates tables in the **current active database**. If they’re missing, it’s possible they were created in master or another unintended database.

1. **Check which database you’re connected to**:  
   Run this query:

sql

Copy

SELECT DB\_NAME() AS CurrentDatabase;

If it returns master, you might have created the tables in the wrong database.

1. **Switch to your intended database**:

sql

Copy

USE YourDatabaseName; GO

Replace YourDatabaseName with the correct name, then rerun the table creation scripts if necessary.

**🔹 Step 2: Check Tables in System Catalog**

Once you’re in the right database, verify that the tables exist:

sql

Copy

SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE = 'BASE TABLE';

If your tables appear, they **exist but may not be under**dbo.

**🔹 Step 3: Check Table Schema**

If you accidentally created tables in a different schema (not dbo), find them using:

sql

Copy

SELECT TABLE\_SCHEMA, TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE = 'BASE TABLE';

If the schema isn’t dbo, reference the full schema in queries:

sql

Copy

SELECT \* FROM YourSchema.YourTable;

**🔹 Step 4: Manually Refresh SSMS Object Explorer**

If you’re checking in **SQL Server Management Studio (SSMS)** and **tables aren’t visible**:

1. Expand your **Database → Tables**.
2. Right-click **Tables → Refresh**.
3. If they still don’t appear, disconnect and reconnect SSMS.

**🔹 Step 5: Recreate Tables in the Correct Database (If Necessary)**

If tables were created in the wrong database, drop them and re-create them in the correct one:

sql

Copy

USE YourDatabaseName; GO DROP TABLE IF EXISTS Projects, Activities, Relationships, Resources, ResourceAssignments, ScenarioData; GO *-- Re-run the table creation scripts here...*

**🚀 Final Check**

Once you’ve fixed the issue, re-run:

sql

Copy

SELECT \* FROM INFORMATION\_SCHEMA.TABLES;

If you see your tables, **you’re good to go!**

Let me know what you find! 🔧🚀

**You said:**

I am in the correct database SQLP6DB, so thats good

**ChatGPT said:**

Great! Since you are in the **correct database (**SQLP6DB**)**, but still don’t see your tables in dbo.TABLES, let’s troubleshoot further.

**🔹 Step 1: Check if Tables Exist in the Database**

Run the following query to confirm if the tables are actually created:

sql

Copy

SELECT TABLE\_SCHEMA, TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE = 'BASE TABLE';

* If your tables appear but **under a different schema (not**dbo**)**, that’s the issue.
* If **no tables appear**, something went wrong during table creation.

**🔹 Step 2: Check If Tables Exist in sys.objects**

Another way to verify table existence:

sql

Copy

SELECT name, schema\_id, type\_desc FROM sys.objects WHERE type = 'U';

* If you see the tables, they exist in SQL Server but may not be visible due to schema differences.

**🔹 Step 3: Ensure You Are Using the**dbo**Schema**

If your tables were created under a different schema, you need to check:

sql

Copy

SELECT s.name AS SchemaName, t.name AS TableName FROM sys.tables t JOIN sys.schemas s ON t.schema\_id = s.schema\_id;

* If tables **aren’t under**dbo, then reference them using SchemaName.TableName.
* If necessary, change them to dbo:

sql

Copy

ALTER SCHEMA dbo TRANSFER YourSchema.YourTable;

**🔹 Step 4: Refresh Table List in SSMS or VS Code**

* If using **SQL Server Management Studio (SSMS)**:
  1. Expand **Databases → SQLP6DB → Tables**.
  2. Right-click **Tables → Refresh**.
* If using **VS Code**:
  1. Click on the **Database Explorer panel**.
  2. Right-click and choose **Refresh Connection**.

**🔹 Step 5: Test Table Accessibility**

Try selecting data from a table:

sql

Copy

SELECT \* FROM dbo.Projects;

* If this fails, try:

sql

Copy

SELECT \* FROM YourSchema.Projects;

* If you get an error, confirm that tables exist using:

sql

Copy

EXEC sp\_MSforeachtable 'SELECT TOP 1 \* FROM ?';

**🚀 Next Steps**

1️⃣ Check **Step 1 & 2** – Do your tables appear?  
2️⃣ If schema mismatch → Use ALTER SCHEMA to fix it.  
3️⃣ If still missing → Try re-running the CREATE TABLE scripts.

Let me know what happens! 🔧🚀

**You said:**

ah, I am thinking...there might of been a step skipped to add the dbo. element

**ChatGPT said:**

Good thinking, mate! If the tables were created **without explicitly specifying**dbo., they might have been assigned to a **different schema** (e.g., your username's default schema).

**🔹 Step 1: Check the Current Schema of Your Tables**

Run this to see where your tables are stored:

sql

Copy

SELECT s.name AS SchemaName, t.name AS TableName FROM sys.tables t JOIN sys.schemas s ON t.schema\_id = s.schema\_id;

* If the SchemaName **isn't**dbo, that’s why the tables don’t appear under dbo.TABLES.

**🔹 Step 2: Move Tables to**dbo**Schema (If Needed)**

If your tables are under a different schema (e.g., YourUsername.Projects), move them to dbo:

sql

Copy

ALTER SCHEMA dbo TRANSFER YourSchema.Projects; ALTER SCHEMA dbo TRANSFER YourSchema.Activities; ALTER SCHEMA dbo TRANSFER YourSchema.Relationships; ALTER SCHEMA dbo TRANSFER YourSchema.Resources; ALTER SCHEMA dbo TRANSFER YourSchema.ResourceAssignments; ALTER SCHEMA dbo TRANSFER YourSchema.ScenarioData;

Replace YourSchema with the actual schema name from **Step 1**.

**🔹 Step 3: Verify Tables Are Now in**dbo

Re-run this check:

sql

Copy

SELECT TABLE\_SCHEMA, TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo';

Now your tables should be listed under dbo.

**🔹 Step 4: Ensure Future Tables Are Created in**dbo

Before creating any new tables, **set the default schema to**dbo:

sql

Copy

ALTER USER YourUsername WITH DEFAULT\_SCHEMA = dbo;

Now, any new tables will be assigned to dbo automatically.

**🚀 Final Test**

Try running:

sql

Copy

SELECT \* FROM dbo.Projects;

* If it works → 🎉 **Problem solved!**
* If still not working → Let me know what error you get, and we'll troubleshoot further.

Nice catch on the missing dbo.—let me know how it goes! 🔧🚀

**You said:**

they do have dbo.Projects etc for each of the tables, ..do we need to restart VS Code before we see them in the table list?

**ChatGPT said:**

Yes! If the tables **have been created under**dbo but aren't showing in the table list, **VS Code might need a refresh**.

**🔹 Step 1: Refresh VS Code SQL Connection**

Try this first:

1. Open VS Code.
2. **Disconnect** from your SQL Server (Ctrl + Shift + P → "MS SQL: Disconnect").
3. **Reconnect** (Ctrl + Shift + P → "MS SQL: Connect").
4. Try running:

sql

Copy

SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'dbo';

If the tables now appear, 🎉 **you're good to go!**

**🔹 Step 2: Manually Refresh Database Explorer in VS Code**

If the tables **still don’t show**:

1. Open the **Database Explorer** panel (if available).
2. Right-click on your **SQL Server connection**.
3. Click **"Refresh"**.

**🔹 Step 3: Restart VS Code (If Needed)**

If refreshing didn’t work:

1. Close **VS Code** completely.
2. Open it again.
3. Reconnect to your SQL database.

Now, check if the tables appear.

**🚀 Final Test**

Try this query to make sure you can **access the tables directly**:

sql

Copy

SELECT \* FROM dbo.Projects;

* If data returns, 🎉 **you're all set!**
* If the tables **still don’t appear**, let me know the exact issue, and we’ll troubleshoot further!

Nice work troubleshooting, mate! 🔧🚀

**You said:**

the tables are in the dbo list, so am happy, and I have figured out how to set up a new query window...so am happy....ready to start taking some data out of p6 and inputting some xls data into the sql model as well.

**ChatGPT said:**

Brilliant, mate! Now that the **tables are set up** and you've got the **query window sorted**, let's move to the **next phase**:

✅ **Extracting Data from Primavera P6 (XER File) and Loading into SQL**  
✅ **Importing Excel Resource Data into SQL**

**🔹 Step 1: Extract Primavera P6 (XER) Data into SQL**

**Option 1: Use P6 XER Export to CSV (Easiest Way)**

1. **In Primavera P6**, go to **File → Export**.
2. Choose **Spreadsheet (XLSX)** → Select **Activities, Relationships, and Resources**.
3. Save the file and open it in Excel to review the structure.

Now, we can **import this CSV data into SQL Server**.

**Option 2: Directly Parse XER into SQL (Advanced)**

If we want to **parse the XER file directly**, we'll need to:

1. Use **a Python script** to extract data from XER.
2. Insert extracted data into SQL tables.

Would you prefer the **CSV method** (faster) or **XER parsing** (more automated)?

**🔹 Step 2: Import Excel Resource Data into SQL**

We need to **map resource availability data** (from Excel) to the ScenarioData table.

**Method 1: Use SQL Server’s Built-in Import Wizard**

1. Open **SQL Server Management Studio (SSMS)**.
2. Right-click your database (SQLP6DB) → **Tasks → Import Data**.
3. Choose **Microsoft Excel** as the data source.
4. Select the **Excel file** with your resource availability data.
5. Map columns to match:
   * **Excel Column → SQL Table (**ScenarioData**)**
   * ResourceID → ResourceID
   * AvailableUnits → AvailableUnits
   * WeekStart → WeekStart
   * WeekEnd → WeekEnd
6. Run the import.

**Method 2: Import Excel Data Using SQL Query**

If the Excel file is structured, we can use **OPENROWSET** to load it:

sql

Copy

SELECT \* INTO ScenarioData FROM OPENROWSET('Microsoft.ACE.OLEDB.12.0', 'Excel 12.0;Database=C:\YourFile.xlsx', 'SELECT \* FROM [Sheet1$]');

**Make sure:**

* Excel file is **saved as**.xlsx.
* SQL Server has **OPENROWSET enabled** (sp\_configure 'Ad Hoc Distributed Queries', 1;).

**🚀 Next Steps**

✅ **Confirm which method you prefer for both P6 and Excel data extraction**.  
✅ I can provide **Python scripts or SQL queries** to automate data loading.  
Let me know your choice, and we'll **get this data moving!** 🚀🔧

**You said:**

i have a connection between P6 and SQL server and its active in VS Code as well, so...we should be able to parse the data straight from P6...am I wrong? you will seriously have to run through this step by step and hold my hand...so please bud, one step at a time ok..

**ChatGPT said:**

You're absolutely right, mate! Since you have a **direct connection between P6 and SQL Server**, we can pull data **straight from P6 into SQL** without needing to export CSVs manually.

**🔹 Plan for Extracting Primavera P6 Data into SQL (Step by Step)**

We'll go through **one step at a time**, making sure each part works before moving to the next.

✅ **Step 1:** Verify Connection to P6 Database in SQL Server  
✅ **Step 2:** Identify the Key P6 Tables for Extraction  
✅ **Step 3:** Run a Simple Query to Check Data  
✅ **Step 4:** Map P6 Data to Our SQL Tables  
✅ **Step 5:** Insert P6 Data into Our SQL Tables

**🔹 Step 1: Verify Connection to P6 Database in SQL Server**

Before pulling any data, let's **make sure SQL Server is properly connected to the P6 database**.

**Run This in VS Code to See Available Databases**

sql

Copy

SELECT name FROM sys.databases;

👉 **What you should see:**

* A list of all databases, including your **P6 database (e.g., PMDB or P6DB)**.

👉 **If your P6 database is missing:**

* Let me know the name of your **P6 database** so we can troubleshoot.

Let me know what this query returns, and we’ll move to **Step 2!** 🚀

**You said:**

SQLP6DB is there and accounted for.

**ChatGPT said:**

Perfect, mate! Since **SQLP6DB** is present, we know the connection is solid. Now, let’s identify the **tables inside the P6 database** so we can extract the right data.

**🔹 Step 2: List All Tables in the P6 Database**

**Run This Query in VS Code to see all tables inside**SQLP6DB**:**

sql

Copy

USE SQLP6DB; GO SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_TYPE = 'BASE TABLE';

👉 **What this does:**

* Lists **all tables** inside SQLP6DB so we can see where P6 stores activities, resources, and relationships.

**🔹 What You Need to Do Next:**

✅ **Run the query** in VS Code.  
✅ **Tell me which tables show up** (or take a screenshot of the table names).

Once I know which tables are available, we’ll move to **Step 3: Extracting Key Data** from P6. 🚀

**You said:**

there are 193 tables which have shown up, not sure how I am going to copy and paste them in here, its not allowing me to select all and copy.

**ChatGPT said:**

No worries, mate! Since **193 tables** have appeared, we just need to find the **key tables** that store the **Activities, Relationships, and Resources** data from P6.

**🔹 Step 3: Identify the Key Primavera P6 Tables**

Since you can’t copy-paste, let’s **filter the tables** to show only the ones we need.

**1️⃣ Find Activity-Related Tables (Tasks)**

Run this query to list **all tables related to activities**:

sql

Copy

SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME LIKE '%TASK%' OR TABLE\_NAME LIKE '%ACTIV%';

👉 **This will show:**

* Tables that store activities (tasks) in P6.

**2️⃣ Find Relationship Tables (Task Dependencies)**

Run this to find **tables related to dependencies (critical path)**:

sql

Copy

SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME LIKE '%REL%' OR TABLE\_NAME LIKE '%DEP%';

👉 **This will show:**

* Tables that define **which task comes before/after another**.

**3️⃣ Find Resource Tables (People, Equipment, Materials)**

Run this to find **tables related to resources**:

sql

Copy

SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME LIKE '%RES%' OR TABLE\_NAME LIKE '%ROLE%';

👉 **This will show:**

* Tables that store **resource assignments**.

**🔹 What You Need to Do Next:**

✅ **Run these three queries one by one** in VS Code.  
✅ **Tell me the exact table names** that show up.

Once we have the right tables, we’ll move to **Step 4: Extracting Data from These Tables**. 🚀

**You said:**

SUMTASK SUMTASKSPREAD TASK TASKACTV TASKDOC TASKFDBK TASKFIN TASKMEMO TASKNOTE TASKPRED TASKPROC TASKRISK TASKRSRC TASKSUM TASK TASKSUMFIN TASKACTV TASKUSER TASKDOC TASKWKSP TASKFDBK TASKFIN TASKMEMO TASKNOTE TASKPRED TASKPROC TASKRISK TASKRSRC TASKSUM TASKSUMFIN TASKUSER TASKWKSP DOCREVIEWTASK DOCREVIEWTASK Activities

**ChatGPT said:**

Great! Based on the tables you've found, we can now focus on **extracting key data** from the Primavera P6 database.

**🔹 Step 4: Identify Key Tables for Extraction**

From your list, here’s what we’ll use:

| **Table Name** | **Purpose** |
| --- | --- |
| TASK | Stores core activity (task) details like ID, name, start/finish dates. |
| TASKPRED | Stores task **predecessors and successors** (for dependencies). |
| TASKRSRC | Stores **resource assignments** linked to tasks. |
| TASKACTV | Stores activity (task) **categories** or activity codes. |

These tables will allow us to **map activities, relationships, and resource assignments** to our SQL model.

**🔹 Step 5: Extract Data from P6 Tables**

Now, let’s pull **some sample data** to verify what’s inside.

**1️⃣ View Activities (Tasks)**

Run this query to extract **basic project tasks**:

sql

Copy

SELECT TOP 10 task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt FROM TASK;

👉 **What this does:**

* Shows **10 tasks** with their **start, end, and float (slack)** times.

**2️⃣ View Task Dependencies (Predecessor-Successor Relationships)**

To see which tasks are **linked together**:

sql

Copy

SELECT TOP 10 task\_id, pred\_task\_id, lag\_hr\_cnt, rel\_type FROM TASKPRED;

👉 **What this does:**

* Shows **task dependencies** (which tasks depend on others).
* rel\_type defines **Finish-Start (FS), Start-Start (SS), Finish-Finish (FF), etc.**

**3️⃣ View Resource Assignments (Who’s Assigned to What Task?)**

To see **which resources are linked to which tasks**:

sql

Copy

SELECT TOP 10 task\_id, res\_id, target\_qty, remain\_qty, budget\_qty FROM TASKRSRC;

👉 **What this does:**

* Shows **which resources** are assigned to each **task**.
* Includes **budgeted & remaining hours** per resource.

**🔹 What You Need to Do Next**

✅ **Run these three queries in VS Code**.  
✅ **Confirm if data shows up correctly**.  
✅ Let me know if we need to **adjust field names** or **filter results further**.

Once we verify this data, we’ll **move to Step 6: Inserting P6 Data into Our SQL Model**! 🚀

**You said:**

hang on, for step 4, those are only the task tables, let me upload the others for you ok?

**ChatGPT said:**

Got it, mate! Upload the **other tables** (Relationships & Resources), and we’ll refine the extraction process accordingly. 🚀

**1/2**

**You said:**

it didnt like the Step 2, query, its shown up an error. Msg 207, Level 16, State 1, Line 5 Invalid column name 'rel\_type'. Total execution time: 00:00:00.003

**ChatGPT said:**

Alright, mate! That error means **the**rel\_type**column doesn’t exist** in TASKPRED. No worries—we just need to check the actual column names inside TASKPRED.

**🔹 Step 1: Check the Column Names in**TASKPRED

Run this query to list all columns in TASKPRED:

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'TASKPRED';

👉 **This will return** the actual column names in TASKPRED, so we can fix the query.

**🔹 Step 2: Adjust the Query Based on Actual Column Names**

Once you see the correct column names, modify the original query to match them.  
For example, if rel\_type doesn’t exist but relationship\_type does, then use:

sql

Copy

SELECT TOP 10 task\_id, pred\_task\_id, lag\_hr\_cnt, relationship\_type *-- Change this based on actual column name* FROM TASKPRED;

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check column names**.  
✅ **Tell me the actual column name for "relationship type"**.  
✅ I’ll adjust the query based on your result!

Let me know what you find, and we’ll fix it together! 🔧🚀

**You said:**

task\_pred\_id task\_id pred\_task\_id proj\_id pred\_proj\_id pred\_type lag\_hr\_cnt update\_date update\_user create\_date create\_user delete\_session\_id delete\_date task\_pred\_id task\_id pred\_task\_id proj\_id pred\_proj\_id pred\_type lag\_hr\_cnt update\_date update\_user create\_date create\_user delete\_session\_id delete\_date

**ChatGPT said:**

Nice work, mate! Based on the **actual column names** in TASKPRED, we need to **update the query** to use pred\_type instead of rel\_type.

**🔹 Step 2 (Fixed Query for Task Dependencies)**

Try this updated query:

sql

Copy

SELECT TOP 10 task\_pred\_id, *-- Unique ID for the dependency* task\_id, *-- Current task* pred\_task\_id, *-- Predecessor task* proj\_id, *-- Project ID* pred\_proj\_id, *-- Predecessor project ID* pred\_type, *-- Type of dependency (FS, SS, FF, SF)* lag\_hr\_cnt *-- Lag time between tasks* FROM TASKPRED;

**🔹 Explanation of Columns:**

* task\_pred\_id → Unique ID for this relationship.
* task\_id → The task that depends on another.
* pred\_task\_id → The task it depends on.
* proj\_id**/**pred\_proj\_id → Ensures tasks are linked to the right project.
* pred\_type → Relationship type (e.g., Finish-to-Start, Start-to-Start).
* lag\_hr\_cnt → Number of hours delay between the tasks.

**🔹 What You Need to Do Next:**

✅ **Run the updated query in VS Code**.  
✅ **Confirm if data looks correct**.  
✅ Let me know if we need any adjustments.

Once this is good, we’ll move to **Step 3: Extracting Resource Assignments**! 🚀

**You said:**

24788 88554 88555 3575 3575 PR\_FS 0.000000 24789 88558 88557 3575 3575 PR\_FS 0.000000 24790 88561 88560 3575 3575 PR\_FS 0.000000 24791 88560 88550 3575 3575 PR\_FS 0.000000 24792 88556 88554 3575 3575 PR\_FS 0.000000 24793 88557 88556 3575 3575 PR\_FS 0.000000 24794 88559 88558 3575 3575 PR\_FS -40.000000 24795 88551 88561 3575 3575 PR\_FS 0.000000 24796 88552 88562 3575 3575 PR\_FS 0.000000 24797 88553 88563 3575 3575 PR\_FS 0.000000

**ChatGPT said:**

Brilliant! That confirms we’re successfully pulling **task dependency data** from TASKPRED. 🎉

Now, let’s move on to **Step 3: Extracting Resource Assignments** from P6.

**🔹 Step 3: Extract Resource Assignments from**TASKRSRC

Since TASKRSRC contains **which resources are assigned to which tasks**, let’s extract the key data.

**Run This Query:**

sql

Copy

SELECT TOP 10 task\_id, *-- Task that has the resource assigned* res\_id, *-- The resource assigned to this task* target\_qty, *-- Total planned hours (or units)* remain\_qty, *-- Remaining hours (or units)* budget\_qty *-- Original budgeted resource allocation* FROM TASKRSRC;

**🔹 Expected Output Example**

You should see something like:

| **task\_id** | **res\_id** | **target\_qty** | **remain\_qty** | **budget\_qty** |
| --- | --- | --- | --- | --- |
| 88554 | 101 | 80 | 40 | 100 |
| 88555 | 102 | 20 | 10 | 25 |

This tells us:

* **Which tasks have resources assigned (**task\_id**)**.
* **Which resources are assigned (**res\_id**)**.
* **How many hours (or units) are planned, remaining, and budgeted**.

**🔹 What You Need to Do Next**

✅ **Run the query in VS Code**.  
✅ **Check if the output looks correct**.  
✅ Let me know if we need any modifications.

Once we confirm this, we’ll **map the data into our SQL model**. 🚀

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 3 Invalid column name 'res\_id'. Msg 207, Level 16, State 1, Line 6 Invalid column name 'budget\_qty'. Total execution time: 00:00:00.015 on running the resource assignment Step 3 query

**ChatGPT said:**

Alright, mate! That error means **the column names in**TASKRSRC**are different** from what we expected. No worries—we just need to check the actual column names and adjust the query.

**🔹 Step 1: Find the Correct Column Names in**TASKRSRC

Run this query to list all columns inside TASKRSRC:

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'TASKRSRC';

👉 **This will return** the actual column names in the TASKRSRC table.

**🔹 Step 2: Adjust the Query Based on Actual Column Names**

Once we get the correct column names, I’ll modify the query to match them.

For example, if res\_id**doesn’t exist** but resource\_id**does**, then we update the query like this:

sql

Copy

SELECT TOP 10 task\_id, resource\_id, *-- Use the actual column name here* target\_qty, remain\_qty, budgeted\_units *-- Use the actual column name here* FROM TASKRSRC;

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check the actual column names**.  
✅ **Tell me what columns are listed for**TASKRSRC.  
✅ I’ll update the query based on your findings!

Let me know what you get, and we’ll fix it together! 🔧🚀

**You said:**

taskrsrc\_id task\_id proj\_id rsrc\_type cost\_qty\_link\_flag role\_id acct\_id rsrc\_id pobs\_id skill\_level pend\_complete\_pct remain\_qty pend\_remain\_qty target\_qty remain\_qty\_per\_hr pend\_act\_reg\_qty target\_lag\_drtn\_hr\_cnt target\_qty\_per\_hr act\_ot\_qty pend\_act\_ot\_qty act\_reg\_qty relag\_drtn\_hr\_cnt ot\_factor cost\_per\_qty target\_cost act\_reg\_cost act\_ot\_cost remain\_cost act\_start\_date act\_end\_date restart\_date reend\_date target\_start\_date target\_end\_date rem\_late\_start\_date rem\_late\_end\_date guid rate\_type act\_this\_per\_cost act\_this\_per\_qty curv\_id rollup\_dates\_flag cost\_per\_qty\_source\_type remain\_crv target\_crv actual\_crv rsrc\_request\_data ts\_pend\_act\_end\_flag prior\_ts\_act\_reg\_qty prior\_ts\_act\_ot\_qty update\_date update\_user create\_date create\_user delete\_session\_id delete\_date taskrsrc\_id task\_id proj\_id rsrc\_type cost\_qty\_link\_flag role\_id acct\_id rsrc\_id pobs\_id skill\_level pend\_complete\_pct remain\_qty pend\_remain\_qty target\_qty remain\_qty\_per\_hr pend\_act\_reg\_qty target\_lag\_drtn\_hr\_cnt target\_qty\_per\_hr act\_ot\_qty pend\_act\_ot\_qty act\_reg\_qty relag\_drtn\_hr\_cnt ot\_factor cost\_per\_qty target\_cost act\_reg\_cost act\_ot\_cost remain\_cost act\_start\_date act\_end\_date restart\_date reend\_date target\_start\_date target\_end\_date rem\_late\_start\_date rem\_late\_end\_date guid rate\_type act\_this\_per\_cost act\_this\_per\_qty curv\_id rollup\_dates\_flag cost\_per\_qty\_source\_type remain\_crv target\_crv actual\_crv rsrc\_request\_data ts\_pend\_act\_end\_flag prior\_ts\_act\_reg\_qty prior\_ts\_act\_ot\_qty update\_date update\_user create\_date create\_user delete\_session\_id delete\_date

**ChatGPT said:**

Awesome! Now that we have the correct column names from TASKRSRC, let's update our **resource assignment query** to match the actual field names.

**🔹 Step 3: Extract Resource Assignments (Fixed Query)**

Run this in VS Code:

sql

Copy

SELECT TOP 10 task\_id, *-- Task that has the resource assigned* rsrc\_id, *-- The resource assigned to this task* target\_qty, *-- Planned total resource units (e.g., hours)* remain\_qty, *-- Remaining resource units (e.g., hours)* target\_cost *-- Total planned cost for the resource* FROM TASKRSRC;

**🔹 Explanation of Columns**

* task\_id → The task the resource is assigned to.
* rsrc\_id → The resource assigned to the task.
* target\_qty → The total planned quantity (hours, units).
* remain\_qty → The remaining resource quantity.
* target\_cost → The total planned cost of the resource.

**🔹 What You Need to Do Next**

✅ **Run the updated query in VS Code**.  
✅ **Check if the data looks correct**.  
✅ Let me know if any adjustments are needed.

Once confirmed, we’ll move to **Step 4: Mapping and Inserting P6 Data into Our SQL Model**! 🚀

**You said:**

88555 4472 51.136250 0.000000 4602.262500 88555 4482 5.000000 0.000000 475.000000 88555 4475 204.545000 0.000000 20454.500000 88556 4475 90.910000 0.000000 9091.000000 88556 4482 90.910000 0.000000 8636.450000 88556 4483 45.455000 0.000000 4545.500000 88556 4472 45.455000 0.000000 4090.950000 88557 4472 34.965385 0.000000 3146.884650 88557 4484 69.930769 0.000000 5944.115365 88557 4475 83.916923 0.000000 8391.692300

**ChatGPT said:**

Perfect! Now we’ve successfully extracted **resource assignments from P6**. 🎉

**🔹 Step 4: Insert P6 Data into Our SQL Model**

Now, we need to **map this data into our SQL tables**.

We'll insert the extracted **Tasks, Relationships, and Resources** into our predefined SQL tables:  
✅ Activities (from TASK)  
✅ Relationships (from TASKPRED)  
✅ ResourceAssignments (from TASKRSRC)

**1️⃣ Insert P6 Activities into Our SQL Table (**Activities**)**

sql

Copy

INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt FROM TASK;

👉 **This will map tasks from**TASK**into our**Activities**table.**

**2️⃣ Insert Task Dependencies into Our SQL Table (**Relationships**)**

sql

Copy

INSERT INTO Relationships (PredID, SuccID, Lag, RelationshipType) SELECT pred\_task\_id, task\_id, lag\_hr\_cnt, pred\_type FROM TASKPRED;

👉 **This maps dependencies from**TASKPRED**into our**Relationships**table.**

**3️⃣ Insert Resource Assignments into Our SQL Table (**ResourceAssignments**)**

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC;

👉 **This links resources to tasks and stores allocation details.**

**🔹 Next Steps**

✅ **Run these queries in VS Code to insert P6 data into SQL.**  
✅ **Confirm that the tables (**Activities**,**Relationships**,**ResourceAssignments**) now have data.**  
✅ Let me know if you hit any issues or need adjustments.

Once this is done, we’ll move to **Step 5: Importing Excel Resource Data into SQL**. 🚀

**You said:**

Started executing query at Line 1 Msg 120, Level 15, State 1, Line 1 The select list for the INSERT statement contains fewer items than the insert list. The number of SELECT values must match the number of INSERT columns. on step 1 for the activities step

**ChatGPT said:**

Good catch, mate! The error means the **number of columns in the**INSERT INTO Activities**statement doesn’t match the columns in the**SELECT**from**TASK. Let’s fix it!

**🔹 Step 1: Check the Column Structure**

First, let’s **double-check the columns in our**Activities**table**.  
Run this in VS Code:

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Activities';

👉 **This will return the actual column names in**Activities, so we can make sure they match the SELECT statement.

**🔹 Step 2: Adjust the**INSERT INTO Activities**Query**

**Possible Issues:**

1️⃣ The Activities table might have **more or fewer columns** than what we're selecting from TASK.  
2️⃣ The column names might need adjustments (e.g., early\_start\_date vs. StartDate).

**Fixed Query for Inserting Activities**

Try this **updated version** that ensures column matching:

sql

Copy

INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, *-- Make sure this column exists in TASK* 0 *-- Default CriticalFlag to 0 (since TASK may not have it)* FROM TASK;

**Key Fixes:**

* **Added**0**as a default value** for CriticalFlag, since TASK may not have that column.
* **Verified all column names align** with TASK.

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check the column names in**Activities**.**  
✅ **If any column names don’t match, tell me, and we’ll adjust the query.**  
✅ **If everything looks good, run the fixed**INSERT**query.**

Once this works, we’ll move to **Step 2: Inserting Relationships (Dependencies) into SQL**. 🚀

**You said:**

ActivityID ProjectID ActivityName StartDate FinishDate Duration CriticalFlag CalendarID for step 1, activities data

**ChatGPT said:**

Great! Now that we have the correct **column names** for Activities, we need to ensure our INSERT INTO**query** correctly matches the columns from TASK.

**🔹 Step 1: Adjust**INSERT INTO Activities**Query**

Since CalendarID exists in Activities, but **TASK does not seem to have a direct equivalent**, we need to: ✔ **Map available columns correctly**  
✔ **Set a default value for**CalendarID

**Try This Fixed Query:**

sql

Copy

INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, *-- Make sure this column exists in TASK* 0, *-- Default CriticalFlag to 0 (since TASK may not have it)* NULL *-- Placeholder for CalendarID (if not available in TASK)* FROM TASK;

**🔹 Explanation of Fixes:**

1️⃣ **Mapped Columns Properly** – Ensured all Activities table columns have a corresponding column from TASK.  
2️⃣ **Added Default for**CriticalFlag – Since TASK may not track it, we set it to 0.  
3️⃣ **Set**CalendarID**to**NULL – If there’s no calendar data in TASK, we allow it to be NULL.

**🔹 What You Need to Do Next**

✅ **Run the updated query in VS Code**.  
✅ **If an error appears, check if**total\_float\_hr\_cnt**exists in**TASK and let me know.  
✅ **Once this works, we’ll move to Step 2 (Insert Task Dependencies)!** 🚀

Let me know how it goes! 🔧

**You said:**

Started executing query at Line 1 Msg 544, Level 16, State 1, Line 1 Cannot insert explicit value for identity column in table 'Activities' when IDENTITY\_INSERT is set to OFF.

**ChatGPT said:**

Good catch! That error means ActivityID**is an identity column**, meaning **SQL Server auto-generates** its values. We don’t need to insert ActivityID manually.

**🔹 Step 1: Fix the**INSERT INTO Activities**Query**

Since ActivityID is auto-generated, **we must remove it from the**INSERT**statement**:

**Try This Updated Query:**

sql

Copy

INSERT INTO Activities (ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, *-- Default CriticalFlag to 0* NULL *-- Placeholder for CalendarID* FROM TASK;

**🔹 Explanation of Fixes:**

1️⃣ **Removed**ActivityID**from**INSERT → SQL Server will now auto-generate it.  
2️⃣ **Ensured Column Count Matches** → Now, SELECT columns exactly match INSERT columns.

**🔹 What You Need to Do Next**

✅ **Run the updated query in VS Code.**  
✅ **If another error appears, let me know.**  
✅ **Once it works, we’ll move to Step 2: Insert Task Dependencies.** 🚀

Let me know how it goes! 🔧

**You said:**

dont we need activity id as the primary key for this table?

**ChatGPT said:**

Yes! ActivityID**is still the primary key**, but since it’s set as an **IDENTITY column**, SQL Server **automatically generates** a unique value for it.

👉 **We do NOT need to insert**ActivityID**manually** because SQL Server will handle it for us.

**🔹 Why We Can’t Insert**ActivityID**Manually**

1️⃣ **IDENTITY columns auto-increment** – SQL Server will assign a unique ID automatically.  
2️⃣ If we try to insert a value into ActivityID manually, **we get an error** (like the one you saw).

**🔹 What If We REALLY Need to Insert**ActivityID**Manually?**

If you absolutely need to insert ActivityID values manually (e.g., migrating data with fixed IDs), you would first need to **enable IDENTITY\_INSERT**:

sql

Copy

SET IDENTITY\_INSERT Activities ON; INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, NULL FROM TASK; SET IDENTITY\_INSERT Activities OFF;

🚨 **Only use this if you MUST retain the exact**task\_id**values from P6!**

**🔹 Recommended Approach (Let SQL Handle**ActivityID**)**

Use this version instead, letting SQL **auto-generate** ActivityID:

sql

Copy

INSERT INTO Activities (ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, NULL FROM TASK;

✅ This keeps ActivityID as the **Primary Key**, but lets SQL **generate values automatically**.

**🔹 What You Need to Do Next**

✅ **Decide if you want to let SQL auto-generate**ActivityID**(Recommended)** or manually insert it.  
✅ **Run the correct query in VS Code.**  
✅ **If it works, we move to Step 2: Insert Task Dependencies!** 🚀

Let me know what you prefer! 🔧

**You said:**

we need to retain it, as it will link to activity Id on other tables.

**ChatGPT said:**

Got it, mate! Since we **need to retain**ActivityID for linking with other tables (like dependencies and resource assignments), we’ll **enable**IDENTITY\_INSERT temporarily to allow manual insertion.

**🔹 Step 1: Insert Activities While Retaining**ActivityID

Use this version of the query:

sql

Copy

SET IDENTITY\_INSERT Activities ON; *-- Allow manual insertion of ActivityID* INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT task\_id, *-- Keep the same ActivityID from P6* proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, *-- Default CriticalFlag to 0* NULL *-- Placeholder for CalendarID* FROM TASK; SET IDENTITY\_INSERT Activities OFF; *-- Disable manual insertion again*

**🔹 Why This Fix Works**

✅ **Retains**ActivityID**from P6 (**task\_id**)** so it stays linked across tables.  
✅ **Prevents errors** by temporarily allowing manual ID insertion.  
✅ **Preserves primary key relationships** for dependencies and resource assignments.

**🔹 What You Need to Do Next**

✅ **Run the updated query in VS Code.**  
✅ **Confirm that**ActivityID**values match**task\_id**from**TASK**.**  
✅ **Once it works, we’ll move to Step 2: Insert Task Dependencies!** 🚀

Let me know how it goes! 🔧

**You said:**

Started executing query at Line 1 Msg 547, Level 16, State 0, Line 3 The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_\_Activitie\_\_Proje\_\_7AC720C5". The conflict occurred in database "SQLP6DB", table "dbo.Projects", column 'ProjectID'. The statement has been terminated. Total execution time: 00:00:00.071

**ChatGPT said:**

Good catch! The error means **some**proj\_id**values in**TASK**don’t exist in the**Projects**table**, causing a **foreign key constraint violation**.

**🔹 Step 1: Check Which**proj\_id**Values Are Causing Issues**

Run this query to see which project IDs in TASK **don’t exist** in Projects:

sql

Copy

SELECT DISTINCT proj\_id FROM TASK WHERE proj\_id NOT IN (SELECT ProjectID FROM Projects);

👉 **This will return** project IDs that exist in TASK but **don’t exist in**Projects.

**🔹 Step 2: Fix the Missing Project IDs**

**Option 1: Insert Missing Projects (Recommended)**

If missing projects are valid and should exist, insert them:

sql

Copy

INSERT INTO Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT DISTINCT proj\_id, 'Unknown Project', *-- Placeholder name* NULL, *-- Placeholder start date* NULL *-- Placeholder end date* FROM TASK WHERE proj\_id NOT IN (SELECT ProjectID FROM Projects);

✅ **This ensures all**proj\_id**values in**TASK**exist in**Projects, avoiding foreign key errors.

**Option 2: Only Insert Activities for Existing Projects**

If some projects in TASK should **not** be included, modify the INSERT INTO Activities query:

sql

Copy

SET IDENTITY\_INSERT Activities ON; INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, NULL FROM TASK WHERE proj\_id IN (SELECT ProjectID FROM Projects); *-- Only insert valid projects* SET IDENTITY\_INSERT Activities OFF;

✅ **This ensures that only activities linked to existing projects are inserted.**

**🔹 What You Need to Do Next**

✅ **Run Step 1** to find missing proj\_id values.  
✅ **Decide whether to insert missing projects (Step 2, Option 1) or filter activities (Step 2, Option 2).**  
✅ **Once fixed, run the**INSERT INTO Activities**query again.**

Let me know what you find! 🚀🔧

**You said:**

3575 3576 3577 3578 3581 3583 3688 3689 3690 3691 3808 3869 3870 3871 3872 3873 3874 3875 3876 3877 3878 3879 3880 3881 3882 3883 3884 3885 3886 3887 3940 4040 4049 4251 4301 4351 4352 4353 4354 4357 4360 4361 4362 4364 4365 4366 4370 4373 4374 4377 4378 4388 4403 4406 4407 4408 4409 4413 4415 4416 4417 4420 4421 4422 4423 4424 4425 4428 4430 4431 4433 4434 4435 4437 4455 4467 4468 4469 4480 4491 4502 4503 4507 4557 from step 1

**ChatGPT said:**

Thanks, mate! Now we know the proj\_id**values in**TASK**that are missing from**Projects, causing the **foreign key constraint error**.

**🔹 Step 2: Fix the Missing Project IDs**

Now, we need to **ensure these projects exist in**Projects before inserting activities.

**✅ Option 1: Insert the Missing Projects (Recommended)**

If these are valid projects that should exist, run this query **to add them to**Projects:

sql

Copy

INSERT INTO Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT DISTINCT proj\_id, 'Unknown Project', *-- Placeholder name* NULL, *-- Placeholder start date* NULL *-- Placeholder end date* FROM TASK WHERE proj\_id NOT IN (SELECT ProjectID FROM Projects);

🔹 This **creates placeholder projects**, allowing activities to be inserted without errors.  
🔹 Later, we can **update these records** with correct project details from P6.

**🚫 Option 2: Skip Activities for Missing Projects**

If these projects **should not** be included in Projects, modify the INSERT INTO Activities query:

sql

Copy

SET IDENTITY\_INSERT Activities ON; INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, NULL FROM TASK WHERE proj\_id IN (SELECT ProjectID FROM Projects); *-- Only insert valid projects* SET IDENTITY\_INSERT Activities OFF;

🔹 This will **only insert activities linked to existing projects** and ignore the missing ones.

**🔹 What You Need to Do Next**

✅ **Decide whether to insert missing projects (Option 1) or filter activities (Option 2).**  
✅ **Run the chosen query in VS Code.**  
✅ **Once done, retry the**INSERT INTO Activities**query.**

Let me know which approach you prefer, and we’ll move forward! 🚀🔧

**You said:**

Started executing query at Line 1 Msg 544, Level 16, State 1, Line 1 Cannot insert explicit value for identity column in table 'Projects' when IDENTITY\_INSERT is set to OFF. Total execution time: 00:00:00.006

**ChatGPT said:**

Ah, mate! That error means ProjectID**is an identity column** in the Projects table, so SQL Server **auto-generates** values for it and won’t let us insert them manually.

Since we **must retain the**proj\_id**from P6**, we need to **enable IDENTITY\_INSERT** temporarily.

**🔹 Step 1: Insert Missing Projects with IDENTITY\_INSERT**

Try this:

sql

Copy

SET IDENTITY\_INSERT Projects ON; *-- Enable manual insertion of ProjectID* INSERT INTO Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT DISTINCT proj\_id, 'Unknown Project', *-- Placeholder name* NULL, *-- Placeholder start date* NULL *-- Placeholder end date* FROM TASK WHERE proj\_id NOT IN (SELECT ProjectID FROM Projects); SET IDENTITY\_INSERT Projects OFF; *-- Disable manual insertion*

**🔹 Why This Works**

✅ **Allows us to insert specific**ProjectID**values** from P6.  
✅ **Preserves the correct relationships between**Projects**and**Activities.  
✅ **Prevents foreign key errors** when inserting activities.

**🔹 What You Need to Do Next**

✅ **Run the updated query in VS Code.**  
✅ **If it works, retry the**INSERT INTO Activities**query.**  
✅ **Let me know if any issues pop up!** 🚀🔧

**You said:**

Started executing query at Line 1 (84 rows affected) Total execution time: 00:00:00.016, i think that worked

**ChatGPT said:**

Boom! 🎉 **84 projects inserted successfully!** That means the missing proj\_id values are now in the Projects table, so we can **retry inserting activities** without foreign key errors.

**🔹 Step 2: Insert Activities Again**

Now, run this query again to insert **activities into our**Activities**table**:

sql

Copy

SET IDENTITY\_INSERT Activities ON; *-- Enable manual insertion of ActivityID* INSERT INTO Activities (ActivityID, ProjectID, ActivityName, StartDate, FinishDate, Duration, CriticalFlag, CalendarID) SELECT task\_id, proj\_id, task\_name, early\_start\_date, early\_end\_date, total\_float\_hr\_cnt, 0, *-- Default CriticalFlag* NULL *-- Placeholder for CalendarID* FROM TASK; SET IDENTITY\_INSERT Activities OFF; *-- Disable manual insertion*

**🔹 What You Need to Do Next**

✅ **Run this query in VS Code.**  
✅ **If it executes successfully, activities should now be in**Activities**.**  
✅ **Confirm by running:**

sql

Copy

SELECT TOP 10 \* FROM Activities;

👉 **If activities show up, we’re all set to move to Step 3: Task Dependencies!** 🚀

Let me know how it goes! 🔧

**You said:**

Open in New Tab 4:29:14 PM Started executing query at Line 1 (15689 rows affected) Total execution time: 00:00:00.043

**ChatGPT said:**

**Boom! 🎉 15,689 activities inserted successfully!** That means the **Activities table is now fully populated** with P6 task data. 🚀

Now, let’s move on to **Step 3: Insert Task Dependencies (Relationships).**

**🔹 Step 3: Insert Task Dependencies into**Relationships

Now that activities are in Activities, we can insert **task dependencies** from TASKPRED.

**Run This Query in VS Code:**

sql

Copy

INSERT INTO Relationships (PredID, SuccID, Lag, RelationshipType) SELECT pred\_task\_id, *-- Predecessor task* task\_id, *-- Successor task* lag\_hr\_cnt, *-- Lag time in hours* pred\_type *-- Relationship type (FS, SS, FF, SF)* FROM TASKPRED;

**🔹 What This Does**

✅ **Links activities together** using predecessor-successor relationships.  
✅ **Preserves the correct scheduling logic** for critical path calculations.  
✅ **Ensures lag time is accounted for in dependencies.**

**🔹 What You Need to Do Next**

✅ **Run the query in VS Code.**  
✅ **If successful, confirm the data with:**

sql

Copy

SELECT TOP 10 \* FROM Relationships;

👉 If dependencies show up, **we’re all set to move to Step 4: Resource Assignments!** 🚀

Let me know how it goes, mate! 🔧

**You said:**

Started executing query at Line 1 Msg 547, Level 16, State 0, Line 1 The INSERT statement conflicted with the CHECK constraint "CK\_\_Relations\_\_Relat\_\_7F8BD5E2". The conflict occurred in database "SQLP6DB", table "dbo.Relationships", column 'RelationshipType'. The statement has been terminated. Total execution time: 00:00:00.027

**ChatGPT said:**

Alright, mate! That error means that some values in pred\_type**(from**TASKPRED**)** don’t match the allowed values in Relationships.RelationshipType.

**🔹 Step 1: Check What Relationship Types Exist in**TASKPRED

Let’s see what **values exist in**pred\_type to find the ones causing the error.

**Run This Query:**

sql

Copy

SELECT DISTINCT pred\_type FROM TASKPRED;

👉 This will show **all relationship types** in TASKPRED.

**🔹 Step 2: Check Allowed Values in**Relationships

In our **SQL model**, we expected **these values** for RelationshipType:

* **FS** (Finish-to-Start)
* **SS** (Start-to-Start)
* **FF** (Finish-to-Finish)
* **SF** (Start-to-Finish)

👉 **If**pred\_type**contains any values that don’t match these, we’ll need to adjust the insert statement.**

**🔹 Step 3: Modify the Insert Query to Handle This**

If pred\_type contains unexpected values, we can **filter them out or map them** before inserting.

**Option 1: Only Insert Allowed Relationship Types**

sql

Copy

INSERT INTO Relationships (PredID, SuccID, Lag, RelationshipType) SELECT pred\_task\_id, task\_id, lag\_hr\_cnt, pred\_type FROM TASKPRED WHERE pred\_type IN ('FS', 'SS', 'FF', 'SF'); *-- Only insert valid values*

🔹 This **prevents invalid relationship types** from being inserted.

**Option 2: Map Unknown Values to a Default Type (If Needed)**

If we find **unexpected values** but want to keep them, we can **map them to a default (e.g., FS)**:

sql

Copy

INSERT INTO Relationships (PredID, SuccID, Lag, RelationshipType) SELECT pred\_task\_id, task\_id, lag\_hr\_cnt, CASE WHEN pred\_type IN ('FS', 'SS', 'FF', 'SF') THEN pred\_type ELSE 'FS' *-- Default unknown values to FS* END FROM TASKPRED;

🔹 This **ensures all values match SQL constraints** and prevents errors.

**🔹 What You Need to Do Next**

✅ **Run Step 1 (**SELECT DISTINCT pred\_type FROM TASKPRED;**)**  
✅ **Tell me what values appear in**pred\_type**.**  
✅ **Choose either Option 1 (filter out invalid types) or Option 2 (map them to a default).**

Once we fix this, we’ll **retry inserting dependencies!** 🚀 Let me know what you find! 🔧

**You said:**

PR\_FF PR\_FS PR\_SS PR\_SF

**ChatGPT said:**

Got it, mate! Your pred\_type values are PR\_FF, PR\_FS, PR\_SS, PR\_SF, but our Relationships table only allows FF, FS, SS, SF.

**🔹 Step 3: Fix the Insert Query (Remove "PR\_")**

We need to **remove "PR\_" from the values before inserting** into Relationships.

**✅ Try This Fixed Query:**

sql

Copy

INSERT INTO Relationships (PredID, SuccID, Lag, RelationshipType) SELECT pred\_task\_id, task\_id, lag\_hr\_cnt, REPLACE(pred\_type, 'PR\_', '') *-- Remove "PR\_" to match allowed values* FROM TASKPRED;

🔹 **This converts**PR\_FS**→**FS**,**PR\_SS**→**SS**, etc.**, ensuring it matches our allowed values.

**🔹 What You Need to Do Next**

✅ **Run the updated query in VS Code.**  
✅ **Check if it executes successfully.**  
✅ **Confirm with:**

sql

Copy

SELECT TOP 10 \* FROM Relationships;

👉 If the relationships show up correctly, we’re ready for **Step 4: Insert Resource Assignments!** 🚀

Let me know how it goes! 🔧

**You said:**

Started executing query at Line 1 (41332 rows affected) Total execution time: 00:00:00.265

**ChatGPT said:**

**Boom! 🎉 41,332 task dependencies inserted successfully!** That means the **Relationships table is now fully populated** with predecessor-successor links. 🚀

Now, let’s move on to **Step 4: Insert Resource Assignments** into SQL.

**🔹 Step 4: Insert Resource Assignments into**ResourceAssignments

This step will link **resources to activities**, storing planned and remaining allocations.

**✅ Run This Query in VS Code:**

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, *-- Activity linked to this resource* rsrc\_id, *-- The resource assigned* target\_qty, *-- Planned resource allocation* remain\_qty, *-- Remaining resource allocation* target\_cost *-- Budgeted resource cost* FROM TASKRSRC;

**🔹 What This Does**

✅ **Links tasks to resources** using task\_id and rsrc\_id.  
✅ **Stores planned (**target\_qty**) and remaining (**remain\_qty**) work hours.**  
✅ **Includes budgeted cost (**target\_cost**).**

**🔹 What You Need to Do Next**

✅ **Run the query in VS Code.**  
✅ **Confirm success by checking the first 10 rows:**

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

👉 If this works, **we're fully done with inserting Primavera P6 data!** 🎉

Let me know how it goes! 🔧🚀

**You said:**

Started executing query at Line 1 Msg 8115, Level 16, State 8, Line 1 Arithmetic overflow error converting numeric to data type numeric. The statement has been terminated. Total execution time: 00:00:00.048

**ChatGPT said:**

Ah, mate! That **arithmetic overflow error** means that one (or more) of the numeric values **from**TASKRSRC is **too large** to fit into the column types in ResourceAssignments.

**🔹 Step 1: Identify Which Column is Causing the Error**

Run this query to check **the maximum values** in TASKRSRC:

sql

Copy

SELECT MAX(target\_qty) AS MaxTargetQty, MAX(remain\_qty) AS MaxRemainQty, MAX(target\_cost) AS MaxTargetCost FROM TASKRSRC;

👉 **This will show** which column has a number **too large** for the destination table (ResourceAssignments).

**🔹 Step 2: Check Your**ResourceAssignments**Table Column Definitions**

Run this query to **see the numeric column types in**ResourceAssignments:

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE, NUMERIC\_PRECISION, NUMERIC\_SCALE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ResourceAssignments';

👉 **If the precision is too low**, we may need to **increase it**.

**🔹 Step 3: Fix the Issue**

Depending on what we find, here are **two possible fixes**:

**✅ Option 1: Increase Column Precision (Recommended)**

If TotalUnits, UnitsPerDay, or BudgetedUnits **have too small a numeric type**, increase their precision:

sql

Copy

ALTER TABLE ResourceAssignments ALTER COLUMN TotalUnits DECIMAL(18,2); *-- Increase if needed* ALTER TABLE ResourceAssignments ALTER COLUMN UnitsPerDay DECIMAL(18,2); ALTER TABLE ResourceAssignments ALTER COLUMN BudgetedUnits DECIMAL(18,2);

🔹 **This ensures large values fit correctly.**

**✅ Option 2: Limit Large Values Before Inserting**

If we **don’t want to change column sizes**, we can **cap extreme values**:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, CAST(target\_qty AS DECIMAL(18,2)), CAST(remain\_qty AS DECIMAL(18,2)), CAST(target\_cost AS DECIMAL(18,2)) FROM TASKRSRC;

🔹 **This forces values into**DECIMAL(18,2)**, rounding them instead of throwing an error.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find the largest values.**  
✅ **Run Step 2 to check column types.**  
✅ **Choose whether to increase precision (Option 1) or cast values (Option 2).**  
✅ **Re-run the**INSERT**query after fixing the issue.**

Let me know what Step 1 returns, and I’ll guide you further! 🚀🔧

**You said:**

85628.000000 85628.000000 154492.800000

**ChatGPT said:**

Alright, mate! The max values you found are:

* target\_qty**(UnitsPerDay)** → **85,628.00**
* remain\_qty**(TotalUnits)** → **85,628.00**
* target\_cost**(BudgetedUnits)** → **154,492.80**

This tells us that **the current column size in**ResourceAssignments**is too small** to handle these values.

**🔹 Step 2: Check Column Definitions**

Run this to see **how much space we currently have**:

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE, NUMERIC\_PRECISION, NUMERIC\_SCALE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ResourceAssignments';

👉 **If the precision is too low (e.g.,**DECIMAL(10,2)**)**, we need to **increase it**.

**🔹 Step 3: Increase Column Precision (Fix the Overflow Error)**

Run this to **expand the numeric storage size** in ResourceAssignments:

sql

Copy

ALTER TABLE ResourceAssignments ALTER COLUMN UnitsPerDay DECIMAL(18,6); ALTER TABLE ResourceAssignments ALTER COLUMN TotalUnits DECIMAL(18,6); ALTER TABLE ResourceAssignments ALTER COLUMN BudgetedUnits DECIMAL(18,6);

🔹 **Why**DECIMAL(18,6)**?**

* Supports **big numbers up to 999 trillion** (999,999,999,999,999.999999).
* Keeps **6 decimal places** for accuracy.
* Solves the **overflow error** while keeping precise values.

**🔹 Step 4: Re-Run the Insert Query**

After adjusting column sizes, try inserting the data again:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC;

**🔹 What You Need to Do Next**

✅ **Run Step 2 to check current column sizes.**  
✅ **Run Step 3 to expand the column sizes.**  
✅ **Run Step 4 to insert data again.**  
✅ **Check if it works, then confirm with:**

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

If it works, **we’re fully done with inserting Primavera P6 data!** 🎉

Let me know if you hit any issues, mate! 🚀🔧

**You said:**

AssignmentID int 10 0 ActivityID int 10 0 ResourceID int 10 0 ProjectID int 10 0 UnitsPerDay decimal 5 2 TotalUnits decimal 10 2 BudgetedUnits decimal 10 2 - This is from Step 2,

**ChatGPT said:**

Got it, mate! Based on your column definitions, the issue is with UnitsPerDay**,**TotalUnits**, and**BudgetedUnits, because their **current size** isn’t large enough to store the big numbers from TASKRSRC (e.g., **85,628.00** and **154,492.80**).

**🔹 Step 3: Increase Column Precision (Fix the Overflow Error)**

Since **your columns are too small**, we need to **increase their precision** to avoid overflow.

**✅ Run This to Expand Column Sizes**

sql

Copy

ALTER TABLE ResourceAssignments ALTER COLUMN UnitsPerDay DECIMAL(12,2); ALTER TABLE ResourceAssignments ALTER COLUMN TotalUnits DECIMAL(12,2); ALTER TABLE ResourceAssignments ALTER COLUMN BudgetedUnits DECIMAL(12,2);

🔹 **Why**DECIMAL(12,2)**?**

* Allows values up to **999,999,999.99** (which easily fits 154,492.80).
* Keeps **2 decimal places** for accuracy.
* Prevents future overflow errors.

**🔹 Step 4: Re-Run the Insert Query**

After adjusting column sizes, try inserting the data again:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC;

**🔹 What You Need to Do Next**

✅ **Run Step 3 to update column sizes.**  
✅ **Run Step 4 to insert the data again.**  
✅ **Check if it works, then confirm with:**

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

If this works, **we’re all set!** 🚀

Let me know if you hit any issues! 🔧

**2/2**

**You said:**

Started executing query at Line 1 Msg 547, Level 16, State 0, Line 1 The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_\_ResourceA\_\_Resou\_\_072CF7AA". The conflict occurred in database "SQLP6DB", table "dbo.Resources", column 'ResourceID'. The statement has been terminated. Total execution time: 00:00:00.132 step 4, caused this error

**ChatGPT said:**

Ah, mate! That **foreign key constraint error** means that some rsrc\_id values in TASKRSRC **do not exist** in the Resources table, causing the insert to fail.

**🔹 Step 1: Identify Missing Resource IDs**

Run this query to find **which**rsrc\_id**values exist in**TASKRSRC**but not in**Resources:

sql

Copy

SELECT DISTINCT rsrc\_id FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources);

👉 This will return **resource IDs that are in**TASKRSRC**but missing from**Resources.

**🔹 Step 2: Fix the Missing Resource IDs**

If the missing rsrc\_id values **should exist**, we need to **insert them into**Resources before inserting into ResourceAssignments.

**✅ Option 1: Insert Missing Resources (Recommended)**

sql

Copy

INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT rsrc\_id, 'Unknown Resource', *-- Placeholder name* NULL, *-- Placeholder role* NULL, *-- Placeholder max units* NULL *-- Placeholder calendar* FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources);

✅ **This will allow all resource assignments to be inserted successfully.**

**🚫 Option 2: Skip Resource Assignments for Missing Resources**

If you **only want to insert valid assignments**, modify the query like this:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IN (SELECT ResourceID FROM Resources); *-- Only insert valid resources*

✅ **This will ignore any resource assignments that reference missing resources.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find missing resource IDs.**  
✅ **Decide whether to insert missing resources (Step 2, Option 1) or filter them out (Option 2).**  
✅ **Re-run the**INSERT INTO ResourceAssignments**query after fixing.**

Let me know what Step 1 returns, and we’ll proceed! 🚀🔧

**You said:**

NULL 1171 1172 1177 1183 1184 1185 1187 1188 1189 1200 1203 1205 1533 1837 4423 4447 4448 4449 4450 4452 4453 4454 4455 4458 4459 4460 4461 4462 4472 4473 4475 4476 4478 4479 4481 4482 4483 4484 6580 6581 6582 6583 6585 6587 6588 6590 6591 6592 6593 6594 6595 6596 6598 6600 6601 6602 6604 6605 6607 6608 6609 6649 6650 6683 6684 6685 6686 6687 6688 6689 6690 6691 6692 6693 6694 6695 6696 6697 6736 6742 6743 6878 6900 6901 6902 6903 6904 6905 6906 6907 6908 6909 6910 6911 6912 6913 6914 6915 6916 6917 6918 6919 6920 6921 6922 6923 6924 6925 6926 6927 6928 6929 6930 6931 6932 6933 6934 6935 6936 6937 6938 6939 6940 6941 6942 6943 6944 6945 6946 6947 6948 6949 6950 6951 6952 6953 6954 6955 6956 6957 6958 6959 6960 6961 6962 6963 6964 6965 6966 6967 6968 6969 6970 6971 6972 6973 6974 6975 6976 6977 6978 6979 6980 6981 6982 6983 6984 6985 6986 6987 6988 6989 6990 6991 6992 6993 6994 6995 6996 6997 6998 6999 7000 7001 7002 7003 7004 7005 7006 7007 7008 7009 7010 7011 7012 7013 7014 7015 7016 7017 7018 7019 7020 7021 7022 7023 7024 7025 7026 7027 7028 7029 7030 7031 7032 7033 7034 7035 7036 7037 7038 7039 7040 7041 7042 7043 7044 7045 7046 7047 7048 7049 7050 7051 7052 7053 7054 7055 7056 7057 7058 7059 from step 1

**ChatGPT said:**

Nice work, mate! This confirms that **these**rsrc\_id**values exist in**TASKRSRC**but are missing from**Resources, which is why the foreign key constraint failed.

**🔹 Step 2: Fix the Missing Resources**

Since these resource IDs should exist, we’ll **insert them into**Resources before inserting assignments.

**✅ Option 1: Insert Missing Resources (Recommended)**

Run this query to **add missing resources** with placeholders:

sql

Copy

SET IDENTITY\_INSERT Resources ON; *-- Allow manual insertion of ResourceID* INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT rsrc\_id, 'Unknown Resource', *-- Placeholder name* NULL, *-- Placeholder role* NULL, *-- Placeholder max units* NULL *-- Placeholder calendar* FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources); SET IDENTITY\_INSERT Resources OFF; *-- Disable manual insertion*

✅ **This will insert all missing resources**, allowing assignments to be added successfully.

**🚫 Option 2: Skip Assignments for Missing Resources**

If we **only want to insert valid assignments**, modify the query like this:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IN (SELECT ResourceID FROM Resources); *-- Only insert valid resources*

✅ **This will ignore any assignments that reference missing resources.**

**🔹 Step 3: Re-Run the Insert Query for Assignments**

Once you’ve fixed the missing resources (Option 1) or filtered out invalid assignments (Option 2), retry inserting assignments:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC;

**🔹 What You Need to Do Next**

✅ **Choose to insert missing resources (Option 1) or skip them (Option 2).**  
✅ **Run the chosen query.**  
✅ **Re-run the**INSERT INTO ResourceAssignments**query.**  
✅ **Check if it works, then confirm with:**

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

If this works, **we’re done with inserting Primavera P6 data!** 🎉

Let me know how it goes! 🚀🔧

**You said:**

Started executing query at Line 1 Msg 515, Level 16, State 2, Line 3 Cannot insert the value NULL into column 'ResourceID', table 'SQLP6DB.dbo.Resources'; column does not allow nulls. INSERT fails. The statement has been terminated. Total execution time: 00:00:00.030

**ChatGPT said:**

Ah, mate! The issue is caused by a NULL**value in**rsrc\_id, and since ResourceID **does not allow NULLs**, the INSERT statement fails.

**🔹 Step 1: Find Where the NULL Values Are Coming From**

Run this query to confirm how many NULL values exist in TASKRSRC:

sql

Copy

SELECT COUNT(\*) AS NullResourceIDs FROM TASKRSRC WHERE rsrc\_id IS NULL;

👉 If the result is **greater than 0**, that means some tasks have **no resource assigned** (NULL).

**🔹 Step 2: Fix the Issue**

We have two ways to handle this:

**✅ Option 1: Exclude NULL**rsrc\_id**Values (Recommended)**

Modify the INSERT query to **only insert non-NULL values**:

sql

Copy

SET IDENTITY\_INSERT Resources ON; *-- Allow manual insertion of ResourceID* INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT rsrc\_id, 'Unknown Resource', *-- Placeholder name* NULL, *-- Placeholder role* NULL, *-- Placeholder max units* NULL *-- Placeholder calendar* FROM TASKRSRC WHERE rsrc\_id IS NOT NULL *-- Exclude NULL values* AND rsrc\_id NOT IN (SELECT ResourceID FROM Resources); SET IDENTITY\_INSERT Resources OFF; *-- Disable manual insertion*

✅ **This prevents the NULL error by skipping NULL**rsrc\_id**values.**

**🚫 Option 2: Replace NULL with a Default Value**

If you want to **assign a default "Unknown" resource**, modify the query:

sql

Copy

SET IDENTITY\_INSERT Resources ON; INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT ISNULL(rsrc\_id, 999999), *-- Replace NULL with 999999 (or another unused ID)* 'Unknown Resource', NULL, NULL, NULL FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources); SET IDENTITY\_INSERT Resources OFF;

✅ **This ensures NULL values are replaced with a default resource (**999999**).**

**🔹 Step 3: Retry the Insert Query for Assignments**

Once resources are fixed, retry inserting assignments:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IS NOT NULL; *-- Ensure no NULLs*

✅ **This ensures that we don’t insert**NULL**resource IDs into**ResourceAssignments**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check how many NULL values exist.**  
✅ **Choose Option 1 (exclude NULLs) or Option 2 (assign default value).**  
✅ **Re-run the**INSERT INTO Resources**query.**  
✅ **Re-run the**INSERT INTO ResourceAssignments**query.**

Let me know how it goes! 🚀🔧

**You said:**

564 from Step 1, counting the number of Null resources in the table

**ChatGPT said:**

Alright, mate! Since **564 NULL values** exist in TASKRSRC, we need to **handle them before inserting into**Resources.

**🔹 Step 2: Fix the NULL Resource Issue**

Since ResourceID **does not allow NULLs**, we have two options:

**✅ Option 1: Ignore NULL Resource Assignments (Recommended)**

If we **don’t need to track tasks without resources**, modify the INSERT to **exclude NULL values**:

sql

Copy

SET IDENTITY\_INSERT Resources ON; INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT rsrc\_id, 'Unknown Resource', NULL, NULL, NULL FROM TASKRSRC WHERE rsrc\_id IS NOT NULL *-- Exclude NULL values* AND rsrc\_id NOT IN (SELECT ResourceID FROM Resources); SET IDENTITY\_INSERT Resources OFF;

✅ **This prevents errors by skipping NULL values.**

**🚫 Option 2: Assign a Default Resource for NULLs**

If you want to **track tasks with missing resources**, assign them to a default "Unknown Resource" with a placeholder ID (999999 or another unused ID).

sql

Copy

SET IDENTITY\_INSERT Resources ON; *-- Insert missing resources* INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT ISNULL(rsrc\_id, 999999), *-- Replace NULL with a default ID* 'Unknown Resource', NULL, NULL, NULL FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources); SET IDENTITY\_INSERT Resources OFF;

✅ **This ensures all tasks get assigned a valid resource.**

**🔹 Step 3: Retry the Insert Query for Assignments**

Once resources are fixed, retry inserting assignments:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, ISNULL(rsrc\_id, 999999), *-- Replace NULLs with the default resource ID* target\_qty, remain\_qty, target\_cost FROM TASKRSRC;

✅ **This ensures that NULL resource IDs don’t break the**INSERT**.**

**🔹 What You Need to Do Next**

✅ **Choose Option 1 (ignore NULLs) or Option 2 (assign default value).**  
✅ **Re-run the**INSERT INTO Resources**query.**  
✅ **Re-run the**INSERT INTO ResourceAssignments**query.**

Let me know which option you prefer and how it goes! 🚀🔧

**You said:**

Started executing query at Line 1 Msg 547, Level 16, State 0, Line 1 The INSERT statement conflicted with the FOREIGN KEY constraint "FK\_\_ResourceA\_\_Resou\_\_072CF7AA". The conflict occurred in database "SQLP6DB", table "dbo.Resources", column 'ResourceID'. The statement has been terminated. Total execution time: 00:00:00.142 - step 3 caused this error...all other steps passed no issue

**ChatGPT said:**

Alright, mate! Since **Steps 1 & 2 worked**, but Step 3 **still failed with a foreign key error**, it means there are still **some**rsrc\_id**values in**TASKRSRC**that are missing from**Resources.

Let’s **double-check and fix this once and for all.**

**🔹 Step 1: Find Which Resource IDs Are Still Missing**

Even after inserting missing resources, some rsrc\_id values might still be missing.

**Run This Query:**

sql

Copy

SELECT DISTINCT rsrc\_id FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources);

👉 This will **return any remaining**rsrc\_id**values** that are causing the INSERT to fail.

**🔹 Step 2: Insert Any Remaining Missing Resources**

If **Step 1 returns missing values**, we need to **insert them into**Resources again.

**✅ Run This to Ensure ALL Missing Resources Are Added**

sql

Copy

SET IDENTITY\_INSERT Resources ON; INSERT INTO Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT DISTINCT rsrc\_id, 'Unknown Resource', NULL, NULL, NULL FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources); SET IDENTITY\_INSERT Resources OFF;

✅ **This ensures every resource assignment has a valid resource ID.**

**🔹 Step 3: Retry the Insert Query for Assignments**

Once **all missing resources are inserted**, retry inserting resource assignments:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IN (SELECT ResourceID FROM Resources); *-- Ensure only valid resources are inserted*

✅ **This guarantees no missing resources exist when inserting assignments.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find any remaining missing resources.**  
✅ **Run Step 2 if missing resources exist.**  
✅ **Retry Step 3 (inserting assignments) and confirm success.**

Once this works, we’re fully done with inserting Primavera P6 data! 🎉

Let me know how it goes! 🚀🔧

**You said:**

rsrc\_id was the only return, from step 1

**ChatGPT said:**

Alright, mate! If **Step 1 only returned**rsrc\_id**(without actual numbers)**, that means **there are no missing resource IDs left** in TASKRSRC.

But **the foreign key error still happened**, meaning something else might be causing it. Let's **troubleshoot further** to get this fixed once and for all.

**🔹 Step 1: Double-Check That All Resource IDs Match**

Since we inserted missing resources, let’s confirm that every resource **in**TASKRSRC**actually exists in**Resources.

**✅ Run This Query to Validate All Resource Assignments**

sql

Copy

SELECT DISTINCT rsrc\_id FROM TASKRSRC WHERE rsrc\_id NOT IN (SELECT ResourceID FROM Resources);

👉 If **this returns no rows**, it means every rsrc\_id in TASKRSRC is present in Resources—which means the issue lies elsewhere.

**🔹 Step 2: Check If There Are Any NULL**rsrc\_id**Values Left**

Since we previously found **564 NULL values**, let’s confirm that none remain:

**✅ Run This Query**

sql

Copy

SELECT COUNT(\*) AS NullResourceCount FROM TASKRSRC WHERE rsrc\_id IS NULL;

👉 If **this returns**0, NULL values are not the issue.  
👉 If **NULL values still exist**, we need to **filter them out again** before inserting assignments.

**🔹 Step 3: Ensure the Foreign Key in**ResourceAssignments**Matches**Resources

The foreign key error might be happening because the ResourceID column **in**ResourceAssignments isn’t exactly matching Resources.ResourceID.

**✅ Check the Foreign Key Constraint**

sql

Copy

SELECT name AS ConstraintName, object\_name(parent\_object\_id) AS TableName, object\_name(referenced\_object\_id) AS ReferencedTableName, type\_desc FROM sys.foreign\_keys WHERE object\_name(parent\_object\_id) = 'ResourceAssignments';

👉 This will confirm if ResourceAssignments.ResourceID is **correctly linked** to Resources.ResourceID.

**🔹 Step 4: Try Inserting Only Valid Assignments Again**

After verifying the above, try this **modified query**, which only inserts valid assignments:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IN (SELECT ResourceID FROM Resources) *-- Only insert valid resources* AND rsrc\_id IS NOT NULL; *-- Ensure no NULL values*

✅ **This ensures we don’t insert invalid or NULL**rsrc\_id**values.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 again to confirm no missing**rsrc\_id**values.**  
✅ **Run Step 2 to check if NULL**rsrc\_id**values still exist.**  
✅ **Run Step 3 to check the foreign key relationship.**  
✅ **If all looks good, re-run the Step 4 query to insert assignments.**

Let me know what Step 1 & 2 return, and we’ll move forward! 🚀🔧

**You said:**

Started executing query at Line 1 (0 rows affected) Total execution time: 00:00:00.049, - for step 1, only returned rsrc\_id again....

**ChatGPT said:**

Alright, mate! Since **Step 1 returned**0 rows affected, that confirms:  
✅ **There are no missing**rsrc\_id**values** in Resources.  
✅ **Every resource ID in**TASKRSRC**exists in**Resources (which means foreign key issues should NOT be happening).

But we **still** got a foreign key constraint error earlier, so let’s dig deeper.

**🔹 Step 2: Check for NULL**rsrc\_id**Values in**TASKRSRC

We previously found **564 NULL values**, so let’s **make sure they’re gone**.

**✅ Run This Query**

sql

Copy

SELECT COUNT(\*) AS NullResourceCount FROM TASKRSRC WHERE rsrc\_id IS NULL;

👉 **If this returns**0, NULL values are not the issue.  
👉 **If NULL values still exist, we need to filter them out in the**INSERT**query.**

**🔹 Step 3: Verify the Foreign Key Relationship in**ResourceAssignments

The **foreign key error** could be caused by **a mismatch in data types** between ResourceAssignments.ResourceID and Resources.ResourceID.

**✅ Run This Query to Check the Foreign Key**

sql

Copy

SELECT name AS ConstraintName, object\_name(parent\_object\_id) AS TableName, object\_name(referenced\_object\_id) AS ReferencedTableName, type\_desc FROM sys.foreign\_keys WHERE object\_name(parent\_object\_id) = 'ResourceAssignments';

👉 This confirms if **ResourceAssignments is correctly linked to Resources**.

Also, check the **data types** of both columns:

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH, NUMERIC\_PRECISION, NUMERIC\_SCALE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ResourceAssignments' AND COLUMN\_NAME = 'ResourceID';

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH, NUMERIC\_PRECISION, NUMERIC\_SCALE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND COLUMN\_NAME = 'ResourceID';

👉 **If the data types don’t match, we need to fix them!**

**🔹 Step 4: Try Inserting Only Valid Assignments Again**

If **everything looks correct** in the previous steps, try inserting only **valid assignments** using this query:

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IN (SELECT ResourceID FROM Resources) *-- Only insert valid resources* AND rsrc\_id IS NOT NULL; *-- Ensure no NULL values*

✅ **This guarantees** we only insert **existing, non-null** resource assignments.

**🔹 What You Need to Do Next**

✅ **Run Step 2 to confirm if NULL**rsrc\_id**values still exist.**  
✅ **Run Step 3 to verify the foreign key and data types.**  
✅ **Run Step 4 to insert valid assignments again.**

Let me know what **Step 2 & Step 3 return**, and we’ll move forward! 🚀🔧

**You said:**

564 - still showing on step 2

**ChatGPT said:**

Alright, mate! Since **Step 2 still shows 564 NULL**rsrc\_id**values**, that confirms the issue:

🔴 **We’re still trying to insert NULL values into**ResourceAssignments.ResourceID**, which is NOT allowed.**

Now, let’s **fix this permanently** by ensuring NULL values are excluded.

**🔹 Step 3: Exclude NULL**rsrc\_id**Values from the Insert**

Since NULL rsrc\_id values exist, we **must filter them out** before inserting into ResourceAssignments.

**✅ Run This Query to Insert Only Valid Assignments**

sql

Copy

INSERT INTO ResourceAssignments (ActivityID, ResourceID, UnitsPerDay, TotalUnits, BudgetedUnits) SELECT task\_id, rsrc\_id, target\_qty, remain\_qty, target\_cost FROM TASKRSRC WHERE rsrc\_id IS NOT NULL *-- Exclude NULL values* AND rsrc\_id IN (SELECT ResourceID FROM Resources); *-- Only insert valid resources*

✅ **This ensures:**  
✔ No NULL values are inserted.  
✔ Only existing ResourceID values from Resources are used.  
✔ Prevents foreign key errors.

**🔹 Step 4: Verify Insert Success**

After running the **Step 3 query**, check if the data is correctly inserted:

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

👉 If results appear, 🎉 **problem solved!**

**🔹 Final Confirmation**

✅ Run Step 3 (Insert Query) again.  
✅ Run Step 4 (SELECT TOP 10 \* FROM ResourceAssignments) to confirm data.  
✅ If all works, we’ve successfully inserted **all P6 data into SQL!** 🚀

Let me know how it goes, mate! 🔧

**You said:**

Started executing query at Line 1 (43954 rows affected) Total execution time: 00:00:00.146 for step 3, now doing step 4

**ChatGPT said:**

**Boom! 🎉 43,954 resource assignments inserted successfully!** That means **all activities, relationships, and resources from Primavera P6 are now fully loaded into SQL!** 🚀

Now, let’s check the data:

**✅ Step 4: Verify the Inserted Data**

Run this query to confirm everything is in ResourceAssignments:

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

👉 If records appear, **we're all set!** 🎉

**🚀 Next Steps**

Now that we have **P6 data successfully loaded into SQL**, we can:  
✅ **Join data with Excel resource availability (ScenarioData table)**  
✅ **Build Power BI dashboards for schedule impact analysis**  
✅ **Implement dynamic scheduling based on real-time resource updates**

Let me know if **Step 4 looks good** and what you’d like to tackle next, mate! 🔧🔥

**You said:**

90356 88555 4472 NULL 51.14 0.00 4602.26 90357 88555 4482 NULL 5.00 0.00 475.00 90358 88555 4475 NULL 204.55 0.00 20454.50 90359 88556 4475 NULL 90.91 0.00 9091.00 90360 88556 4482 NULL 90.91 0.00 8636.45 90361 88556 4483 NULL 45.46 0.00 4545.50 90362 88556 4472 NULL 45.46 0.00 4090.95 90363 88557 4472 NULL 34.97 0.00 3146.88 90364 88557 4484 NULL 69.93 0.00 5944.12 90365 88557 4475 NULL 83.92 0.00 8391.69

**ChatGPT said:**

Perfect! 🎉 Your ResourceAssignments table is now **fully populated**, but I see that the **ProjectID column is showing NULL values**.

**🔹 Step 5: Fix the Missing**ProjectID**in**ResourceAssignments

Since ProjectID should be present, we need to **update it using data from**Activities (since Activities links ActivityID to ProjectID).

**✅ Run This Query to Update**ProjectID

sql

Copy

UPDATE RA SET RA.ProjectID = A.ProjectID FROM ResourceAssignments RA JOIN Activities A ON RA.ActivityID = A.ActivityID;

✅ **This will:**  
✔ Fill in missing ProjectID values by looking them up in Activities.  
✔ Ensure all resource assignments are correctly linked to their projects.

**🔹 Step 6: Verify the Fix**

After running the update, check if ProjectID is now populated:

sql

Copy

SELECT TOP 10 \* FROM ResourceAssignments;

👉 If ProjectID values are now present, **problem solved!** 🎉

**🚀 Next Steps**

✅ **Run Step 5 to fix missing**ProjectID**values.**  
✅ **Run Step 6 to verify the update worked.**  
✅ **Once confirmed, we’re ready to move to Power BI dashboard setup!** 🚀

Let me know how it goes! 🔧🔥

**You said:**

90356 88555 4472 3575 51.14 0.00 4602.26 90357 88555 4482 3575 5.00 0.00 475.00 90358 88555 4475 3575 204.55 0.00 20454.50 90359 88556 4475 3575 90.91 0.00 9091.00 90360 88556 4482 3575 90.91 0.00 8636.45 90361 88556 4483 3575 45.46 0.00 4545.50 90362 88556 4472 3575 45.46 0.00 4090.95 90363 88557 4472 3575 34.97 0.00 3146.88 90364 88557 4484 3575 69.93 0.00 5944.12 90365 88557 4475 3575 83.92 0.00 8391.69

**ChatGPT said:**

**Boom! 🎉 The**ProjectID**values are now correctly updated!** That means all data from **Primavera P6 is fully loaded and structured in SQL.** 🚀

**🔹 Next Steps: Integrating with Power BI**

Now that the **database is set up**, we can:  
✅ **Connect SQL Server to Power BI**  
✅ **Build dynamic dashboards to analyze resource impact**  
✅ **Incorporate real-time Excel data for scenario analysis**

Would you like me to guide you through **connecting Power BI to SQL** and creating a **resource impact dashboard**? 🔥💡

**You said:**

morning Jarvis, welcome to day 2, of Project Hack 24. we need to continue with the tables in SQL?...are you down with that.

**ChatGPT said:**

**Good morning, mate!** ☕ Welcome to **Day 2 of Project Hack 24!** 🚀 Ready to tackle more SQL and push this project forward.

**🔹 Quick Recap from Yesterday**

✔ **Primavera P6 data (Activities, Relationships, Resource Assignments) loaded into SQL**  
✔ **Foreign key issues fixed, and missing data filled in**  
✔ **ProjectIDs successfully linked in**ResourceAssignments

**🔹 What’s Next for SQL Today?**

Are we:  
✅ Adding **new tables** (e.g., scenario planning, real-time resource updates)?  
✅ Refining **existing tables** (indexes, constraints, normalization)?  
✅ Preparing data for **Power BI visualization**?

Let me know the priority, and we’ll get cracking! 🔧🔥

**You said:**

we need to create the tables for Projects, and Resources first. I need to talk to you about the Scenario data table, ...then we can look at next steps to get the required output to Excel, etc.

**ChatGPT said:**

**Got it, mate!** 🎯 Let’s start by creating the **Projects** and **Resources** tables first, then we’ll chat about the **ScenarioData** table before moving on to Excel outputs.

**🔹 Step 1: Create the**Projects**Table**

We’ll store **all project-level details** here, ensuring they are properly linked to activities and resources.

sql

Copy

CREATE TABLE Projects ( ProjectID INT PRIMARY KEY IDENTITY(1,1), *-- Auto-incremented primary key* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL );

✅ **Why?**  
✔ Each project gets a **unique ID**.  
✔ Stores project **names and dates** for scheduling insights.

**🔹 Step 2: Create the**Resources**Table**

This will store **all available resources (people, equipment, materials, etc.)**.

sql

Copy

CREATE TABLE Resources ( ResourceID INT PRIMARY KEY IDENTITY(1,1), *-- Auto-incremented unique ID* ResourceName NVARCHAR(255) NOT NULL, Role NVARCHAR(100) NULL, *-- The type of resource (Engineer, Operator, etc.)* MaxUnits DECIMAL(10,2) NULL, *-- Maximum available hours per day* CalendarID INT NULL );

✅ **Why?**  
✔ Each resource gets a **unique ID**.  
✔ Links resources to **assignments and scheduling**.

**🔹 Next Steps**

1️⃣ **Run these**CREATE TABLE**statements in VS Code**.  
2️⃣ **Confirm the tables appear in SQL Server (**SELECT \* FROM Projects;**and**SELECT \* FROM Resources;**).**  
3️⃣ **Let me know when you're ready to chat about**ScenarioData**.**

Once these are in place, we can discuss how to structure **ScenarioData for real-time adjustments**. 🔧🚀

**You said:**

P6, has its own Project ID, ...but we will also be implementing to use Workpack as well, to compare different variations of the same Workpack ID....are you ok with that

**ChatGPT said:**

Got it, mate! **P6 has its own**ProjectID, but we’re also introducing **Workpack IDs** to compare different versions of the same Workpack.

This means our Projects table needs to:  
✔ **Retain P6’s**ProjectID for integration  
✔ **Include**WorkpackID to compare different variations  
✔ **Support multiple project structures**

**🔹 Step 1: Modify the**Projects**Table to Include**WorkpackID

We’ll add a WorkpackID**column** and ensure **each project version is linked to its Workpack**.

**✅ Updated**Projects**Table**

sql

Copy

CREATE TABLE Projects ( ProjectID INT PRIMARY KEY, *-- P6's unique project ID* WorkpackID NVARCHAR(50) NOT NULL, *-- ID for comparing variations* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL );

✅ **Why This Works**  
✔ ProjectID **matches P6's data**  
✔ WorkpackID **groups variations of the same Workpack**  
✔ Allows **multiple projects under one Workpack for analysis**

**🔹 Step 2: Modify Related Tables to Support**WorkpackID

Since we’re adding **Workpack tracking**, should we also:  
✅ **Link**WorkpackID**to**Activities?  
✅ **Include**WorkpackID**in**ResourceAssignments to track resource usage per Workpack?

**🔹 What You Need to Do Next**

✅ **Run the updated**CREATE TABLE Projects**query**  
✅ **Let me know if**WorkpackID**should also go into**Activities**&**ResourceAssignments

Once confirmed, we’ll move to **ScenarioData design**! 🔧🚀

**You said:**

we created a Project code called "Workpack ID", you might have to search in the Database to find what the project called it....This will allow us to compare multiple project scenarios with different project ids but the same workpack id....do you follow this?

**ChatGPT said:**

**Got it, mate!** ✅ The **Workpack ID** exists in P6 as a **Project Code** rather than a column in the Projects table.

So, to correctly link Workpacks across different ProjectIDs, we need to:  
1️⃣ **Find where P6 stores Workpack IDs** in the database.  
2️⃣ **Extract Workpack IDs and link them to Projects**.  
3️⃣ **Ensure multiple**ProjectID**s can be grouped under the same**WorkpackID**for comparison**.

**🔹 Step 1: Find the Table Storing Workpack IDs**

We’ll search the P6 database for where **Project Codes** (including Workpack ID) are stored.

**✅ Run This Query to Find Relevant Tables**

sql

Copy

SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%workpack%' OR COLUMN\_NAME LIKE '%proj\_code%' OR COLUMN\_NAME LIKE '%code%';

👉 This will **list all tables that contain "workpack" or project codes**.

**🔹 Step 2: Extract Workpack IDs from P6**

Once we find the right table, we’ll **extract the Workpack IDs** and link them to Projects.

**✅ Example Query (Adjust Once We Find the Correct Table)**

sql

Copy

SELECT p.ProjectID, c.WorkpackID, p.ProjectName, p.StartDate, p.EndDate FROM Projects p JOIN Some\_Project\_Code\_Table c ON p.ProjectID = c.ProjectID WHERE c.CodeType = 'Workpack ID';

👉 This will show how Workpack IDs are linked to **different ProjectIDs**.

**🔹 Step 3: Modify the**Projects**Table to Include WorkpackID**

Once we confirm the correct data source, we’ll **update our Projects table** so it includes WorkpackID for scenario comparisons.

**✅ Updated Projects Table**

sql

Copy

CREATE TABLE Projects ( ProjectID INT PRIMARY KEY, *-- P6's unique project ID* WorkpackID NVARCHAR(50) NULL, *-- Workpack ID from P6 project codes* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL );

✅ This allows us to **group multiple**ProjectID**s under one**WorkpackID**for comparisons**.

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find the table storing Workpack IDs**  
✅ **Let me know what table stores them**  
✅ **Then we’ll extract and update the**Projects**table with Workpack IDs**

Once that’s done, we’ll move to **ScenarioData design**! 🚀🔧

**You said:**

ITERGOAL JOBSVC ITERGOAL OPENING\_ACTVCODE OPENING\_ACTVCODE LOCATION LOCATION LOCATION NONWORK RSRC JOBSVC LOCATION LOCATION LOCATION PREFER NONWORK RSRCHOUR PROJECT PROJECT PROJECT PROJECT PROJISSU PROJRISK PROJRISK PROJTHRS PROJWBS PREFER TASK TASK TASKACTV TASKACTV RSRC PRMQUEUE RSRCHOUR PROJECT PROJECT PROJECT PROJECT TASK TASK TASKACTV TASKACTV ACTVCODE ACTVCODE ACTVCODE ACTVCODE ACTVTYPE ACTVTYPE ACTVTYPE TIMESHT BRE\_REGISTRY BUDGCHNG TIMESHT TSAUDIT UDFCODE UDFCODE UDFCODE UDFCODE UDFCODE UDFCODE UDFTYPE UDFTYPE DLTACTV DLTACTV UDFVALUE UDFVALUE PROJISSU ACTVCODE ACTVCODE ACTVCODE ACTVCODE ACTVTYPE ACTVTYPE ACTVTYPE PROJRISK PROJRISK DOCSTAT BRE\_REGISTRY USESSION BUDGCHNG PROJTHRS PROJWBS USESSAUD DOCSTAT USESSION

**ChatGPT said:**

Alright, mate! Based on the table list, I see multiple instances of PROJECT and ACTVCODE, which are the most likely places where Workpack IDs might be stored.

**🔹 Step 1: Check Where Workpack IDs Are Stored**

We need to run queries on PROJECT and ACTVCODE to find out if they contain the **Workpack ID**.

**✅ Check If Workpack ID Exists in**PROJECT

Run this to see the **columns** in PROJECT:

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT';

👉 **Look for anything related to**WorkpackID**,**proj\_code**, or**UDF**(User Defined Field).**

If WorkpackID or a similar column exists, check its values:

sql

Copy

SELECT TOP 10 \* FROM PROJECT;

👉 **Look for a column containing Workpack IDs.**

**✅ Check If Workpack ID Exists in**ACTVCODE

Since **P6 sometimes stores project codes in**ACTVCODE, run this:

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ACTVCODE';

If the table looks relevant, check its values:

sql

Copy

SELECT TOP 10 \* FROM ACTVCODE;

👉 Look for **any field that resembles Workpack IDs**.

**🔹 Step 2: Extract Workpack IDs and Link to**Projects

Once we confirm **where Workpack IDs are stored**, we’ll **join that data to**Projects and modify our table.

**Updated**Projects**Table (Once We Identify Workpack Source)**

sql

Copy

CREATE TABLE Projects ( ProjectID INT PRIMARY KEY, *-- P6's unique project ID* WorkpackID NVARCHAR(50) NULL, *-- Workpack ID from P6 project codes* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL );

**🔹 What You Need to Do Next**

✅ **Run Step 1 (**SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT'**&**ACTVCODE**)**  
✅ **Let me know if any column looks like**WorkpackID  
✅ **Once identified, we’ll link it to**Projects

This will allow **Workpack-based scenario comparisons!** 🚀 Let me know what you find! 🔧

**You said:**

proj\_id fy\_start\_month\_num chng\_eff\_cmp\_pct\_flag rsrc\_self\_add\_flag allow\_complete\_flag rsrc\_multi\_assign\_flag checkout\_flag project\_flag step\_complete\_flag cost\_qty\_recalc\_flag sum\_only\_flag batch\_sum\_flag name\_sep\_char def\_complete\_pct\_type proj\_short\_name acct\_id orig\_proj\_id source\_proj\_id base\_type\_id clndr\_id sum\_base\_proj\_id task\_code\_base task\_code\_step priority\_num wbs\_max\_sum\_level risk\_level strgy\_priority\_num last\_checksum critical\_drtn\_hr\_cnt def\_cost\_per\_qty last\_recalc\_date plan\_start\_date plan\_end\_date scd\_end\_date add\_date sum\_data\_date last\_tasksum\_date fcst\_start\_date def\_duration\_type task\_code\_prefix guid def\_qty\_type add\_by\_name web\_local\_root\_path proj\_url def\_rate\_type act\_this\_per\_link\_flag def\_task\_type act\_pct\_link\_flag add\_act\_remain\_flag critical\_path\_type task\_code\_prefix\_flag def\_rollup\_dates\_flag rem\_target\_link\_flag reset\_planned\_flag allow\_neg\_act\_flag rsrc\_id msp\_managed\_flag msp\_update\_actuals\_flag checkout\_date checkout\_user\_id sum\_assign\_level last\_fin\_dates\_id use\_project\_baseline\_flag last\_baseline\_update\_date ts\_rsrc\_vw\_compl\_asgn\_flag ts\_rsrc\_mark\_act\_finish\_flag ts\_rsrc\_vw\_inact\_actv\_flag cr\_external\_key apply\_actuals\_date description intg\_proj\_type matrix\_id location\_id control\_updates\_flag hist\_interval hist\_level update\_date update\_user create\_date create\_user delete\_session\_id delete\_date proj\_id fy\_start\_month\_num chng\_eff\_cmp\_pct\_flag rsrc\_self\_add\_flag allow\_complete\_flag rsrc\_multi\_assign\_flag checkout\_flag project\_flag step\_complete\_flag cost\_qty\_recalc\_flag sum\_only\_flag batch\_sum\_flag name\_sep\_char def\_complete\_pct\_type proj\_short\_name acct\_id orig\_proj\_id source\_proj\_id base\_type\_id clndr\_id sum\_base\_proj\_id task\_code\_base task\_code\_step priority\_num wbs\_max\_sum\_level risk\_level strgy\_priority\_num last\_checksum critical\_drtn\_hr\_cnt def\_cost\_per\_qty last\_recalc\_date plan\_start\_date plan\_end\_date scd\_end\_date add\_date sum\_data\_date last\_tasksum\_date fcst\_start\_date def\_duration\_type task\_code\_prefix guid def\_qty\_type add\_by\_name web\_local\_root\_path proj\_url def\_rate\_type act\_this\_per\_link\_flag def\_task\_type act\_pct\_link\_flag add\_act\_remain\_flag critical\_path\_type task\_code\_prefix\_flag def\_rollup\_dates\_flag rem\_target\_link\_flag reset\_planned\_flag allow\_neg\_act\_flag rsrc\_id msp\_managed\_flag msp\_update\_actuals\_flag checkout\_date checkout\_user\_id sum\_assign\_level last\_fin\_dates\_id use\_project\_baseline\_flag last\_baseline\_update\_date ts\_rsrc\_vw\_compl\_asgn\_flag ts\_rsrc\_mark\_act\_finish\_flag ts\_rsrc\_vw\_inact\_actv\_flag cr\_external\_key apply\_actuals\_date description intg\_proj\_type matrix\_id location\_id control\_updates\_flag hist\_interval hist\_level update\_date update\_user create\_date create\_user delete\_session\_id delete\_date

**ChatGPT said:**

Alright, mate! Looking at the PROJECT table, **I don’t see an obvious**WorkpackID**field**, but I have a couple of ideas on where it might be stored.

Since Workpack ID is a **Project Code** in P6, it’s likely stored in **another table**, such as:  
1️⃣ ACTVCODE (if Workpack ID is stored as an activity code)  
2️⃣ UDFVALUE (if Workpack ID is a user-defined field)

**🔹 Step 1: Check If Workpack ID Exists in**ACTVCODE

Since **activity codes** can be assigned to projects, Workpack ID might be in ACTVCODE.

**✅ Run This Query to List Columns in**ACTVCODE

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ACTVCODE';

👉 **Look for any column related to project codes.**

If it looks relevant, check the values:

sql

Copy

SELECT TOP 10 \* FROM ACTVCODE;

👉 **See if Workpack IDs appear in the data.**

**🔹 Step 2: Check If Workpack ID Exists in**UDFVALUE

If Workpack ID is stored as a **User-Defined Field (UDF)** in P6, it could be in UDFVALUE.

**✅ Run This Query to List Columns in**UDFVALUE

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'UDFVALUE';

If it looks relevant, check the values:

sql

Copy

SELECT TOP 10 \* FROM UDFVALUE;

👉 **Look for a column containing Workpack IDs.**

**🔹 Step 3: Extract Workpack ID and Link to**Projects

Once we find where **Workpack ID is stored**, we’ll **join it to**Projects so we can use it for scenario comparisons.

**🔹 What You Need to Do Next**

✅ **Run Step 1 (**ACTVCODE**check) and Step 2 (**UDFVALUE**check).**  
✅ **Let me know if you find Workpack ID in either table.**  
✅ **Once identified, we’ll update**Projects**to store Workpack IDs.**

This will allow us to **group multiple**ProjectID**s under the same**WorkpackID**for comparisons!** 🚀 Let me know what you find! 🔧

**You said:**

actv\_code\_id actv\_code\_type\_id seq\_num short\_name parent\_actv\_code\_id actv\_code\_name color update\_date update\_user create\_date create\_user delete\_session\_id delete\_date actv\_code\_id actv\_code\_type\_id seq\_num short\_name parent\_actv\_code\_id actv\_code\_name color update\_date update\_user create\_date create\_user delete\_session\_id delete\_date

**ChatGPT said:**

Nice work, mate! The ACTVCODE table **contains activity codes**, but I don’t see an obvious reference to projects or Workpack IDs yet.

**🔹 Step 1: Check If**ACTVCODE**Links to Projects**

Since ACTVCODE stores codes, we need to check if it links to **projects** or **activities**.

**✅ Run This Query to Find Related Tables**

sql

Copy

SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME = 'actv\_code\_id';

👉 **This will show other tables that reference**ACTVCODE**.**

If a PROJECT**or**TASK**table appears**, that means activity codes are assigned to projects or tasks.

**🔹 Step 2: Check the Values in**ACTVCODE

Now, let’s see if **Workpack IDs exist** in ACTVCODE.

**✅ Run This Query**

sql

Copy

SELECT TOP 20 \* FROM ACTVCODE;

👉 **Look at**actv\_code\_name**and**short\_name – do they contain Workpack IDs?

If Workpack IDs exist, we need to **link this table to projects**.

**🔹 Step 3: Check If Workpack ID is in**UDFVALUE

If ACTVCODE doesn’t store Workpack IDs, they might be in **User-Defined Fields (UDF)**.

**✅ Run This Query to See UDF Table Structure**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'UDFVALUE';

👉 **Look for fields related to**proj\_id**or**workpack**.**

Then, check the values:

sql

Copy

SELECT TOP 20 \* FROM UDFVALUE;

👉 **Look for Workpack IDs.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check where**actv\_code\_id**is used.**  
✅ **Run Step 2 to see if**ACTVCODE**stores Workpack IDs.**  
✅ **If no Workpack ID appears, check**UDFVALUE**.**

Once we confirm where Workpack IDs are stored, we’ll **link them to Projects and move forward!** 🚀 Let me know what you find! 🔧

**You said:**

1328 91 0 Expense NULL Expense NULL NULL NULL NULL NULL NULL NULL 1329 91 100 Capital NULL Capital NULL NULL NULL NULL NULL NULL NULL 1338 95 300 Dependency NULL Dependency NULL NULL NULL NULL NULL NULL NULL 1339 95 100 Req Late NULL Requirements late NULL NULL NULL NULL NULL NULL NULL 1340 95 200 Test Data NULL Testing Data NULL NULL NULL NULL NULL NULL NULL 1388 95 0 Staff Avail NULL Staff Availability NULL NULL NULL NULL NULL NULL NULL 1449 127 0 Y NULL Yes NULL 2010-08-06 16:23:25.000 admin1 NULL NULL NULL NULL 1457 130 0 Planned NULL Planned Work NULL 2007-06-29 18:05:36.000 exec 2007-06-29 18:05:36.000 exec NULL NULL 1500 142 0 REL1 NULL Release 1 NULL 2008-01-02 15:50:58.000 admin 2008-01-02 15:50:14.000 admin NULL NULL 1501 142 100 REL2 NULL Release 2 NULL 2008-01-02 15:51:00.000 admin 2008-01-02 15:50:16.000 admin NULL NULL 1502 142 200 REL3 NULL Release 3 NULL 2008-01-02 15:51:02.000 admin 2008-01-02 15:50:16.000 admin NULL NULL 1503 142 300 REL4 NULL Release 4 NULL 2008-01-02 15:51:04.000 admin 2008-01-02 15:50:16.000 admin NULL NULL 1504 142 400 REL5 NULL Release 5 NULL 2008-01-02 15:51:06.000 admin 2008-01-02 15:50:17.000 admin NULL NULL 1505 142 500 REL6 NULL Release 6 NULL 2008-01-02 15:51:09.000 admin 2008-01-02 15:50:34.000 admin NULL NULL 1506 142 600 REL7 NULL Release 7 NULL 2008-01-02 15:51:11.000 admin 2008-01-02 15:50:34.000 admin NULL NULL 1507 142 700 REL8 NULL Release 8 NULL 2008-01-02 15:51:14.000 admin 2008-01-02 15:50:34.000 admin NULL NULL 1508 142 800 REL9 NULL Release 9 NULL 2008-01-02 15:51:17.000 admin 2008-01-02 15:50:46.000 admin NULL NULL 1509 130 100 Rework NULL Rework NULL 2008-01-09 12:55:57.000 admin 2008-01-09 12:55:48.000 admin NULL NULL 1511 143 0 Key MS NULL Key Milestones NULL 2010-01-15 17:32:18.000 admin 2008-01-09 12:56:44.000 admin NULL NULL 1518 147 10 ENG NULL Engineering Department NULL 2008-05-13 15:21:32.000 admin 2008-05-13 15:21:32.000 admin NULL NULL

**ChatGPT said:**

Alright, mate! Based on the ACTVCODE data you provided, I don’t see an obvious WorkpackID, but these codes could still be linked to projects elsewhere in P6.

Since Workpack ID is a **Project Code**, we need to check **if**ACTVCODE**links to**PROJECT through another table.

**🔹 Step 1: Find Where**ACTVCODE**Links to**PROJECT

We need to check if project activity codes (including Workpack ID) are stored in a **separate mapping table**.

**✅ Run This Query to Find Tables Referencing**actv\_code\_id

sql

Copy

SELECT TABLE\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME = 'actv\_code\_id';

👉 This will show **which tables store activity codes** and how they might relate to projects.

If any PROJECT**-related tables appear**, check the values:

sql

Copy

SELECT TOP 10 \* FROM [TABLE\_NAME\_FROM\_ABOVE\_QUERY];

👉 Look for proj\_id alongside actv\_code\_id to see if they are linked.

**🔹 Step 2: Check If Workpack ID is in**UDFVALUE

If Workpack ID is stored as a **User-Defined Field (UDF)** in P6, it could be in UDFVALUE.

**✅ Run This Query to See UDF Table Structure**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'UDFVALUE';

If relevant, check the values:

sql

Copy

SELECT TOP 20 \* FROM UDFVALUE;

👉 **Look for fields that reference Workpack IDs.**

**🔹 Step 3: Extract Workpack ID and Link to**Projects

Once we confirm where Workpack IDs are stored, we’ll **join that data to**Projects and modify our table.

**🔹 What You Need to Do Next**

✅ **Run Step 1 (**actv\_code\_id**references) to see where**ACTVCODE**is used.**  
✅ **If no link appears, check**UDFVALUE**for Workpack IDs.**  
✅ **Once identified, we’ll update**Projects**to store Workpack IDs.**

This will allow us to **group multiple**ProjectID**s under the same**WorkpackID**for scenario comparisons!** 🚀 Let me know what you find! 🔧

**You said:**

If i only just added the project code to the EPS node in P6, is there a refresh SQL model, or something like that, ...or maybe refresh P6, by pressing F5, to update the table lists

**ChatGPT said:**

Ah, gotcha, mate! Since you **just added the project code to the EPS node in P6**, the database **might not have updated yet**.

**🔹 Step 1: Refresh P6 to Ensure Changes Are Saved**

Try these **P6 refresh methods** before checking SQL:

**✅ Option 1: Manually Refresh P6 Data**

1️⃣ Press **F5** to refresh the view in P6.  
2️⃣ If that doesn’t work, **log out and back into P6**.  
3️⃣ Try running **"Tools → Schedule (F9)"** to update project data.

**✅ Option 2: Force a Database Refresh in P6**

1️⃣ In P6, go to **Admin → Admin Preferences → Services**.  
2️⃣ Check if the **"Apply Actuals" or "Summarization" services** are scheduled to run soon.  
3️⃣ If possible, **manually trigger a summarization run**.

**🔹 Step 2: Check If SQL Has Updated**

Once you’ve refreshed P6, check if the **Workpack ID now appears in SQL**.

**✅ Run This Query to See Recent Updates in**PROJECT

sql

Copy

SELECT TOP 10 \* FROM PROJECT ORDER BY update\_date DESC;

👉 **If your change was saved, your project should have a recent**update\_date**timestamp.**

**✅ Check If the Workpack ID Now Exists in**ACTVCODE**or**UDFVALUE

Run the same **queries from before** to check if new Workpack data is now available in SQL.

**🔹 Step 3: If SQL Still Doesn’t Show the Data**

If SQL **doesn’t show your changes**, you might need to:  
1️⃣ **Wait for P6's scheduled background jobs** to sync the data.  
2️⃣ **Restart the P6 database service** (if you have admin access).  
3️⃣ **Manually re-enter the project code** to ensure it was saved correctly.

**🔹 What You Need to Do Next**

✅ **Try refreshing P6 (F5, log out/in, or schedule refresh).**  
✅ **Run the SQL queries again to see if Workpack ID is now in the database.**  
✅ **Let me know if SQL still doesn’t show the update.**

If it's still missing, we’ll explore **forcing a manual SQL refresh** or checking for **delayed database synchronization.** 🚀 Let me know how it goes! 🔧

**You said:**

give me some code that can look for EC515 in Workpack ID, from the Project codes on the EPS level

**ChatGPT said:**

Alright, mate! Since **Workpack ID (EC515)** was added as a **Project Code at the EPS level**, we need to:  
✅ **Find where project codes are stored in SQL**  
✅ **Check if**EC515**appears in those tables**

**🔹 Step 1: Identify Where Project Codes Are Stored**

We suspect **project codes** are in ACTVCODE, UDFVALUE, or another project-related table.

**✅ Run This Query to Search All Tables for "EC515"**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%code%' OR COLUMN\_NAME LIKE '%workpack%';

👉 **This will show where project codes or Workpack IDs might be stored.**

**🔹 Step 2: Search for EC515 in Likely Tables**

Once we identify the right table (e.g., ACTVCODE, UDFVALUE, PROJECT), search for EC515.

**✅ Check if EC515 is in**ACTVCODE

sql

Copy

SELECT \* FROM ACTVCODE WHERE actv\_code\_name LIKE '%EC515%' OR short\_name LIKE '%EC515%';

👉 **If**EC515**is found**, it means Workpack IDs are stored in ACTVCODE.

**✅ Check if EC515 is in**UDFVALUE**(User-Defined Fields)**

sql

Copy

SELECT \* FROM UDFVALUE WHERE udf\_text LIKE '%EC515%';

👉 **If**EC515**appears here, Workpack ID is stored as a UDF.**

**🔹 Step 3: Link Workpack ID to Projects**

Once we confirm where **Workpack IDs are stored**, we’ll **join that data to**Projects for proper tracking.

**✅ Join Workpack ID with**Projects

sql

Copy

SELECT p.ProjectID, p.ProjectName, c.actv\_code\_name AS WorkpackID FROM PROJECT p JOIN ACTVCODE c ON p.ProjectID = c.actv\_code\_id WHERE c.actv\_code\_name LIKE '%EC515%';

👉 This links **Workpack ID from**ACTVCODE**to**Projects.

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find tables with Workpack codes.**  
✅ **Run Step 2 queries to check where**EC515**is stored.**  
✅ **If found, run Step 3 to link Workpack IDs to**Projects**.**

Once we confirm the correct data source, we’ll **update our SQL model to store Workpack IDs permanently!** 🚀 Let me know what you find! 🔧

**You said:**

ITERGOAL status\_code JOBSVC status\_code ITERGOAL status\_code OPENING\_ACTVCODE actv\_code\_id OPENING\_ACTVCODE parent\_actv\_code\_id LOCATION country\_code LOCATION postal\_code LOCATION state\_code NONWORK nonwork\_code RSRC employee\_code JOBSVC status\_code LOCATION country\_code LOCATION postal\_code LOCATION state\_code PREFER task\_code\_len NONWORK nonwork\_code RSRCHOUR status\_code PROJECT task\_code\_base PROJECT task\_code\_prefix PROJECT task\_code\_prefix\_flag PROJECT task\_code\_step PROJISSU status\_code PROJRISK risk\_code PROJRISK status\_code PROJTHRS status\_code PROJWBS status\_code PREFER task\_code\_len TASK status\_code TASK task\_code TASKACTV actv\_code\_id TASKACTV actv\_code\_type\_id RSRC employee\_code PRMQUEUE status\_code RSRCHOUR status\_code PROJECT task\_code\_base PROJECT task\_code\_prefix PROJECT task\_code\_prefix\_flag PROJECT task\_code\_step TASK status\_code TASK task\_code TASKACTV actv\_code\_id TASKACTV actv\_code\_type\_id ACTVCODE actv\_code\_id ACTVCODE actv\_code\_name ACTVCODE actv\_code\_type\_id ACTVCODE parent\_actv\_code\_id ACTVTYPE actv\_code\_type ACTVTYPE actv\_code\_type\_id ACTVTYPE actv\_code\_type\_scope TIMESHT status\_code BRE\_REGISTRY status\_code BUDGCHNG status\_code TIMESHT status\_code TSAUDIT ts\_status\_code UDFCODE parent\_udf\_code\_id UDFCODE udf\_code\_id UDFCODE udf\_code\_name UDFCODE parent\_udf\_code\_id UDFCODE udf\_code\_id UDFCODE udf\_code\_name UDFTYPE udf\_code\_short\_len UDFTYPE udf\_code\_short\_len DLTACTV actv\_code\_id DLTACTV parent\_actv\_code\_id UDFVALUE udf\_code\_id UDFVALUE udf\_code\_id PROJISSU status\_code ACTVCODE actv\_code\_id ACTVCODE actv\_code\_name ACTVCODE actv\_code\_type\_id ACTVCODE parent\_actv\_code\_id ACTVTYPE actv\_code\_type ACTVTYPE actv\_code\_type\_id ACTVTYPE actv\_code\_type\_scope PROJRISK risk\_code PROJRISK status\_code DOCSTAT doc\_status\_code BRE\_REGISTRY status\_code USESSION hard\_drive\_code BUDGCHNG status\_code PROJTHRS status\_code PROJWBS status\_code USESSAUD hard\_drive\_code DOCSTAT doc\_status\_code USESSION hard\_drive\_code

**ChatGPT said:**

Nice work, mate! Based on the table scan, we have **two strong candidates** for where Workpack IDs might be stored:

1️⃣ ACTVCODE → Stores actv\_code\_name, which could include Workpack IDs.  
2️⃣ UDFVALUE → Stores udf\_code\_id, meaning Workpack ID might be a **User-Defined Field (UDF)** in P6.

**🔹 Step 1: Check If EC515 (Workpack ID) Exists in**ACTVCODE

Since ACTVCODE stores activity codes, let’s check if EC515 appears there.

**✅ Run This Query**

sql

Copy

SELECT \* FROM ACTVCODE WHERE actv\_code\_name LIKE '%EC515%' OR actv\_code\_id IN ( SELECT actv\_code\_id FROM TASKACTV );

👉 **If EC515 appears**, it means Workpack IDs are stored in ACTVCODE.

**🔹 Step 2: Check If EC515 Exists in**UDFVALUE

If Workpack IDs are stored as **User-Defined Fields (UDFs)**, we need to check UDFVALUE.

**✅ Run This Query**

sql

Copy

SELECT \* FROM UDFVALUE WHERE udf\_code\_id IN ( SELECT udf\_code\_id FROM UDFCODE WHERE udf\_code\_name LIKE '%Workpack%' );

👉 **If Workpack ID is here, it’s a UDF.**

**🔹 Step 3: Extract and Link Workpack IDs to**Projects

Once we find **where Workpack IDs are stored**, we’ll **link them to Projects** so we can compare different scenarios.

**✅ Join Workpack ID with**Projects**(If Found in**ACTVCODE**)**

sql

Copy

SELECT p.ProjectID, p.ProjectName, c.actv\_code\_name AS WorkpackID FROM PROJECT p JOIN ACTVCODE c ON p.ProjectID = c.actv\_code\_id WHERE c.actv\_code\_name LIKE '%EC515%';

👉 This links Workpack ID from ACTVCODE to Projects.

**🔹 What You Need to Do Next**

✅ **Run Step 1 (**ACTVCODE**check) and Step 2 (**UDFVALUE**check) to locate Workpack IDs.**  
✅ **Let me know if**EC515**is found in either table.**  
✅ **Once identified, we’ll update**Projects**to store Workpack IDs permanently!**

This will allow us to **group multiple**ProjectID**s under the same**WorkpackID**for scenario comparisons!** 🚀 Let me know what you find! 🔧

**You said:**

no, thats not it....hmm...how can I find out, what the field is actually called in P6

**ChatGPT said:**

Good thinking, mate! Since you **just added the Workpack ID as a Project Code at the EPS level**, we need to **find out exactly what P6 calls this field in the database**.

**🔹 Step 1: Find the Project Code Name in P6**

To identify the correct field in SQL, first check **what P6 calls it in the UI**:

**✅ In P6, Do This:**

1️⃣ **Go to Enterprise → Project Codes**  
2️⃣ Look for the **Workpack ID category**  
3️⃣ Note the **exact name of the field** (e.g., "Workpack Code", "WPID", "Workpack Identifier")

**🔹 Step 2: Search SQL for the Exact Field Name**

Once we know the **name from P6**, we’ll search for it in the database.

**✅ Search for the Exact Field in SQL**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%workpack%' OR COLUMN\_NAME LIKE '%proj\_code%' OR COLUMN\_NAME LIKE '%code%';

👉 This will return **the exact table and column name** that stores the Workpack ID.

**🔹 Step 3: Extract Workpack IDs from the Correct Table**

Once we find the right table, we’ll **pull Workpack IDs** and link them to Projects.

**✅ Example Query (Adjust Once We Find the Correct Table)**

sql

Copy

SELECT p.ProjectID, c.[CorrectFieldName], p.ProjectName, p.StartDate, p.EndDate FROM PROJECT p JOIN [CorrectTable] c ON p.ProjectID = c.ProjectID WHERE c.[CorrectFieldName] LIKE '%EC515%';

👉 This will confirm Workpack IDs are correctly stored.

**🔹 What You Need to Do Next**

✅ **Check P6 (Enterprise → Project Codes) for the exact field name**  
✅ **Run Step 2 in SQL to locate the correct table and column**  
✅ **Let me know what P6 calls the field, and we’ll pull the data!**

Once we identify the field, we can **link Workpack IDs to Projects and move forward!** 🚀 Let me know what you find! 🔧

**You said:**

I need a field called Project Code Name, and in there you will find Workpack ID, and then the associated field will hold EC515 or DHBL

**ChatGPT said:**

**Got it, mate!** ✅ We need to find a field called **"Project Code Name"**, which contains **"Workpack ID"**, and the associated value (like **EC515** or **DHBL**) will be stored alongside it.

**🔹 Step 1: Find Where "Project Code Name" is Stored in SQL**

Since **Project Codes** are part of P6’s EPS structure, they’re likely stored in a **code-related table**, such as PROJECT, ACTVCODE, or UDFVALUE.

**✅ Run This Query to Find the Table Containing "Project Code Name"**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%proj\_code%' OR COLUMN\_NAME LIKE '%project\_code%' OR COLUMN\_NAME LIKE '%code\_name%';

👉 This will **list tables** that contain project codes.

**🔹 Step 2: Search for "Workpack ID" in the Identified Table**

Once we identify the correct table, check if it contains **"Workpack ID"** as a project code name.

**✅ Try This Query to Find "Workpack ID"**

sql

Copy

SELECT \* FROM [CorrectTableFromStep1] WHERE [CorrectColumn] LIKE '%Workpack ID%';

👉 **If "Workpack ID" is found**, the table is correct!

**🔹 Step 3: Retrieve Workpack IDs (EC515, DHBL, etc.)**

Once we find the table, we’ll **extract Workpack IDs** and link them to projects.

**✅ Pull Workpack IDs and Associated Projects**

sql

Copy

SELECT p.ProjectID, pc.[Project Code Name], pc.[Associated Value] AS WorkpackID FROM PROJECT p JOIN [CorrectTable] pc ON p.ProjectID = pc.ProjectID WHERE pc.[Project Code Name] = 'Workpack ID';

👉 This will **list projects alongside their Workpack IDs (EC515, DHBL, etc.).**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find which table stores "Project Code Name".**  
✅ **Run Step 2 to check if "Workpack ID" appears.**  
✅ **Run Step 3 to extract Workpack IDs and link them to projects.**

Once we confirm this, we’ll **update the SQL model to store Workpack IDs permanently!** 🚀 Let me know what you find! 🔧

**You said:**

ACTVCODE actv\_code\_name UDFCODE udf\_code\_name UDFCODE udf\_code\_name ACTVCODE actv\_code\_name

**ChatGPT said:**

Nice work, mate! Based on the results, the **most likely locations for "Project Code Name" (Workpack ID)** are:  
1️⃣ ACTVCODE.actv\_code\_name  
2️⃣ UDFCODE.udf\_code\_name

Now, let’s confirm which one actually holds **"Workpack ID"** as a category, and then pull the associated values (EC515, DHBL, etc.).

**🔹 Step 1: Find If "Workpack ID" Exists in**ACTVCODE

Since ACTVCODE**stores activity codes**, we need to check if **"Workpack ID" is listed as a code name.**

**✅ Run This Query**

sql

Copy

SELECT \* FROM ACTVCODE WHERE actv\_code\_name LIKE '%Workpack%';

👉 **If "Workpack ID" appears**, we’ve found the right table!

**🔹 Step 2: Check If "Workpack ID" Exists in**UDFCODE

If ACTVCODE doesn’t store **Workpack ID**, it might be in UDFCODE.

**✅ Run This Query**

sql

Copy

SELECT \* FROM UDFCODE WHERE udf\_code\_name LIKE '%Workpack%';

👉 **If "Workpack ID" appears here, then Workpack IDs are stored as UDFs.**

**🔹 Step 3: Extract Workpack IDs for Projects**

Once we find **where Workpack IDs are stored**, we’ll **link them to Projects**.

**✅ If Workpack ID is in**ACTVCODE

sql

Copy

SELECT p.ProjectID, a.actv\_code\_name AS WorkpackID, p.ProjectName FROM PROJECT p JOIN ACTVCODE a ON p.ProjectID = a.actv\_code\_id WHERE a.actv\_code\_name = 'Workpack ID';

**✅ If Workpack ID is in**UDFCODE

sql

Copy

SELECT p.ProjectID, u.udf\_code\_name AS WorkpackID, p.ProjectName FROM PROJECT p JOIN UDFCODE u ON p.ProjectID = u.udf\_code\_id WHERE u.udf\_code\_name = 'Workpack ID';

**🔹 What You Need to Do Next**

✅ **Run Step 1 (**ACTVCODE**check) and Step 2 (**UDFCODE**check) to locate "Workpack ID".**  
✅ **Once identified, run Step 3 to pull Workpack IDs and link them to projects.**

This will allow us to **compare multiple project scenarios using Workpack IDs!** 🚀 Let me know what you find! 🔧

**You said:**

its not in either mate, I am lost

**ChatGPT said:**

No worries, mate! If **"Workpack ID"** isn’t in ACTVCODE or UDFCODE, it’s likely stored in **another project-related table**. Let’s systematically find it.

**🔹 Step 1: Find Where "Workpack ID" is Stored in SQL**

Since **P6 treats project codes separately**, we need to **search for "Workpack ID" across all tables**.

**✅ Run This Query to Search Every Table for "Workpack ID"**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%code%' OR COLUMN\_NAME LIKE '%workpack%' OR COLUMN\_NAME LIKE '%proj%';

👉 **This will list all possible tables that contain project-related codes.**

**🔹 Step 2: Search for "Workpack ID" in the Identified Tables**

Once Step 1 shows relevant tables, pick **the most relevant one** and search for "Workpack ID".

**✅ Try Searching the Most Likely Table (Example)**

sql

Copy

SELECT \* FROM [MostLikelyTable] WHERE [MostLikelyColumn] LIKE '%Workpack ID%';

👉 If we find "Workpack ID" here, that’s the table we need!

**🔹 Step 3: Search for "EC515" or "DHBL" Directly in SQL**

Since Workpack IDs (like EC515) are stored as values, let’s **search for them across all tables**.

**✅ Run This Query to Find Where "EC515" Exists**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%value%' OR COLUMN\_NAME LIKE '%text%' OR COLUMN\_NAME LIKE '%name%';

👉 **This will list tables that might store EC515 or DHBL as values.**

Once you find a table with relevant column names, search for **EC515**:

sql

Copy

SELECT \* FROM [TableFromAbove] WHERE [ColumnFromAbove] LIKE '%EC515%';

👉 If this returns results, **we’ve found the Workpack ID storage!**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find project-related code tables.**  
✅ **Run Step 2 to search for "Workpack ID".**  
✅ **Run Step 3 to search for "EC515" or "DHBL".**  
✅ **Tell me what you find, and we’ll extract Workpack IDs correctly!**

Once we locate the data, we’ll **link it to**Projects**and move forward!** 🚀 Let me know how it goes, mate! 🔧

**You said:**

give me code to find "Risk Rating" in Project codes, in the P6 database, which the table contained here, will then contain Workpack ID

**ChatGPT said:**

Alright, mate! Since **"Risk Rating"** is a **Project Code in P6**, and we know the table containing it will also have **Workpack ID**, let’s find it systematically.

**🔹 Step 1: Find Which Table Stores "Risk Rating"**

We’ll **search all tables** that contain the **Project Codes** for "Risk Rating".

**✅ Run This Query to Find "Risk Rating" in P6 Database**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%code%' OR COLUMN\_NAME LIKE '%proj%' OR COLUMN\_NAME LIKE '%risk%';

👉 **This will list all possible tables that store project codes like "Risk Rating".**

**🔹 Step 2: Search for "Risk Rating" in the Identified Table**

Once Step 1 finds a relevant table, **search for "Risk Rating" as a project code name**.

**✅ Try Searching the Most Likely Table (Example)**

sql

Copy

SELECT \* FROM [MostLikelyTable] WHERE [MostLikelyColumn] LIKE '%Risk Rating%';

👉 If "Risk Rating" is found, that’s the table we need!

**🔹 Step 3: Retrieve Workpack IDs from the Same Table**

Since the table storing "Risk Rating" **also contains Workpack IDs**, we can extract both.

**✅ Extract Workpack IDs Alongside Risk Ratings**

sql

Copy

SELECT p.ProjectID, pc.[Project Code Name], pc.[Associated Value] AS WorkpackID FROM PROJECT p JOIN [CorrectTable] pc ON p.ProjectID = pc.ProjectID WHERE pc.[Project Code Name] IN ('Risk Rating', 'Workpack ID');

👉 **This will return all Workpack IDs (EC515, DHBL, etc.) and Risk Ratings linked to projects.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find the table storing "Risk Rating".**  
✅ **Run Step 2 to confirm "Risk Rating" exists in that table.**  
✅ **Run Step 3 to extract both Workpack IDs and Risk Ratings.**

Once we confirm the right table, we’ll **update SQL to store Workpack IDs permanently**! 🚀 Let me know what you find, mate! 🔧

**You said:**

scratch that, lets look for Strategic Objective in Project Code, on the EPS node of P6 Database

**ChatGPT said:**

Got it, mate! Since **"Strategic Objective"** is a **Project Code in P6 (EPS level)**, we need to find where P6 stores **Project Codes** at the EPS level. Once we find this, we’ll also locate **Workpack ID** in the same table.

**🔹 Step 1: Find Which Table Stores "Strategic Objective"**

We need to **search all project code-related tables** for "Strategic Objective".

**✅ Run This Query to Identify the Table**

sql

Copy

SELECT TABLE\_NAME, COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE COLUMN\_NAME LIKE '%code%' OR COLUMN\_NAME LIKE '%proj%' OR COLUMN\_NAME LIKE '%strategy%' OR COLUMN\_NAME LIKE '%objective%';

👉 **This will list all tables that might store project codes like "Strategic Objective".**

**🔹 Step 2: Search for "Strategic Objective" in the Identified Table**

Once Step 1 finds a relevant table, **search for "Strategic Objective" in it**.

**✅ Try Searching the Most Likely Table (Example)**

sql

Copy

SELECT \* FROM [MostLikelyTable] WHERE [MostLikelyColumn] LIKE '%Strategic Objective%';

👉 If "Strategic Objective" is found, that’s the table we need!

**🔹 Step 3: Retrieve Workpack IDs from the Same Table**

Since the table storing "Strategic Objective" **also contains Workpack IDs**, we can extract both.

**✅ Extract Workpack IDs Alongside Strategic Objectives**

sql

Copy

SELECT p.ProjectID, pc.[Project Code Name], pc.[Associated Value] AS WorkpackID FROM PROJECT p JOIN [CorrectTable] pc ON p.ProjectID = pc.ProjectID WHERE pc.[Project Code Name] IN ('Strategic Objective', 'Workpack ID');

👉 **This will return all Workpack IDs (EC515, DHBL, etc.) and Strategic Objectives linked to projects.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find the table storing "Strategic Objective".**  
✅ **Run Step 2 to confirm "Strategic Objective" exists in that table.**  
✅ **Run Step 3 to extract both Workpack IDs and Strategic Objectives.**

Once we confirm the right table, we’ll **update SQL to store Workpack IDs permanently**! 🚀 Let me know what you find, mate! 🔧

**You said:**

PFLOAD\_UACCESS\_VIEW proj\_id ISSUHIST proj\_id PFLOAD\_IP\_UACCESS\_VIEW proj\_id ITERDAYS proj\_id WBSSTEP proj\_id ITERDAYS proj\_id ITERGOAL status\_code ResourceAssignments ProjectID WKFLTMPL project\_flag RPT proj\_id WORKFLOW existing\_project\_flag WORKFLOW proj\_id JOBSVC status\_code ITERGOAL status\_code OPENING\_ACTVCODE actv\_code\_id OPENING\_ACTVCODE parent\_actv\_code\_id LOCATION country\_code LOCATION postal\_code LOCATION state\_code RPTBATCH proj\_id MEMOTYPE proj\_flag WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_WEEK proj\_id WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_WEEK project\_summ\_qty NONWORK nonwork\_code WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_MONTH proj\_id WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_MONTH project\_summ\_qty WORKSPACE proj\_id RSRC employee\_code WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_QTR proj\_id WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_QTR project\_summ\_qty JOBSVC status\_code WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_YEAR proj\_id WBSRSRC\_ASSGN\_BY\_PROJ\_BY\_YEAR project\_summ\_qty PCATTYPE max\_proj\_catg\_wt PCATTYPE proj\_catg\_short\_len PCATTYPE proj\_catg\_type PCATTYPE proj\_catg\_type\_id PCATTYPE proj\_catg\_type\_wt WBSRSRC\_ASSGN\_BY\_RSRC\_BY\_WEEK project\_summ\_qty PCATUSER proj\_catg\_id LOCATION country\_code LOCATION postal\_code LOCATION state\_code WBSRSRC\_ASSGN\_BY\_RSRC\_BY\_MONTH project\_summ\_qty PCATVAL parent\_proj\_catg\_id PCATVAL proj\_catg\_id PCATVAL proj\_catg\_name PCATVAL proj\_catg\_short\_name PCATVAL proj\_catg\_type\_id PCATVAL proj\_catg\_wt MEMOTYPE proj\_flag WBSRSRC\_ASSGN\_BY\_RSRC\_BY\_QTR project\_summ\_qty PFOLIO closed\_proj\_flag PFOLIO whatif\_proj\_flag WBSRSRC\_ASSGN\_BY\_RSRC\_BY\_YEAR project\_summ\_qty WBSRSRC\_ASSGN\_BY\_ROLE\_BY\_WEEK project\_summ\_qty WBSRSRC\_ASSGN\_BY\_ROLE\_BY\_MONTH project\_summ\_qty WBSRSRC\_ASSGN\_BY\_ROLE\_BY\_QTR project\_summ\_qty PREFER proj\_short\_len PREFER projinit\_admin\_user\_id PREFER task\_code\_len WBSRSRC\_ASSGN\_BY\_ROLE\_BY\_YEAR project\_summ\_qty NONWORK nonwork\_code WBSRSRC\_ASSGN\_BY\_PROJRSRC\_WEEK PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJRSRC\_WEEK PROJECT\_SUMM\_QTY WBSRSRC\_ASSGN\_BY\_PROJRSRC\_MON PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJRSRC\_MON PROJECT\_SUMM\_QTY RSRCHOUR proj\_id RSRCHOUR status\_code WBSRSRC\_ASSGN\_BY\_PROJRSRC\_QTR PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJRSRC\_QTR PROJECT\_SUMM\_QTY OBSPROJ proj\_id WBSRSRC\_ASSGN\_BY\_PROJRSRC\_YEAR PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJRSRC\_YEAR PROJECT\_SUMM\_QTY PROJCOST proj\_id PCATTYPE max\_proj\_catg\_wt PCATTYPE proj\_catg\_short\_len PCATTYPE proj\_catg\_type PCATTYPE proj\_catg\_type\_id PCATTYPE proj\_catg\_type\_wt WBSRSRC\_ASSGN\_BY\_PROJROLE\_WEEK PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJROLE\_WEEK PROJECT\_SUMM\_QTY PROJECT intg\_proj\_type PROJECT orig\_proj\_id PROJECT proj\_id PROJECT proj\_short\_name PROJECT proj\_url PROJECT project\_flag PROJECT use\_project\_baseline\_flag PROJECT source\_proj\_id PROJECT sum\_base\_proj\_id PROJECT task\_code\_base PROJECT task\_code\_prefix PROJECT task\_code\_prefix\_flag PROJECT task\_code\_step WBSRSRC\_ASSGN\_BY\_PROJROLE\_MON PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJROLE\_MON PROJECT\_SUMM\_QTY PROJEST proj\_est\_id PROJEST proj\_id WBSRSRC\_ASSGN\_BY\_PROJROLE\_QTR PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJROLE\_QTR PROJECT\_SUMM\_QTY PROJFUND proj\_fund\_id PROJFUND proj\_id WBSRSRC\_ASSGN\_BY\_PROJROLE\_YEAR PROJ\_ID WBSRSRC\_ASSGN\_BY\_PROJROLE\_YEAR PROJECT\_SUMM\_QTY PROJISSU base\_proj\_id PROJISSU proj\_id PROJISSU status\_code PCATUSER proj\_catg\_id WBSRSRC\_ASSGN\_SUMMARY\_BY\_MON proj\_id WBSRSRC\_ASSGN\_SUMMARY\_BY\_MON project\_summ\_qty PROJPCAT proj\_catg\_id PROJPCAT proj\_catg\_type\_id PROJPCAT proj\_id PCATVAL parent\_proj\_catg\_id PCATVAL proj\_catg\_id PCATVAL proj\_catg\_name PCATVAL proj\_catg\_short\_name PCATVAL proj\_catg\_type\_id PCATVAL proj\_catg\_wt WBSRSRC\_ASSGN\_SUMMARY\_BY\_QTR proj\_id WBSRSRC\_ASSGN\_SUMMARY\_BY\_QTR project\_summ\_qty PROJPROP proj\_id WBSRSRC\_ASSGN\_SUMMARY\_BY\_WEEK proj\_id WBSRSRC\_ASSGN\_SUMMARY\_BY\_WEEK project\_summ\_qty PROJRISK proj\_id PROJRISK risk\_code PROJRISK status\_code SCENPROJ proj\_id PFOLIO closed\_proj\_flag PFOLIO whatif\_proj\_flag WBSRSRC\_ASSGN\_SUMMARY\_BY\_YEAR proj\_id WBSRSRC\_ASSGN\_SUMMARY\_BY\_YEAR project\_summ\_qty PROJTHRS proj\_id PROJTHRS status\_code WBSRSRC\_ASSN\_BY\_PROJ\_BY\_FP proj\_id WBSRSRC\_ASSN\_BY\_PROJ\_BY\_FP proj\_summ\_qty PROJWBS proj\_id PROJWBS proj\_node\_flag PROJWBS status\_code WBSRSRC\_ASSN\_BY\_PROJROLE\_BY\_FP proj\_id WBSRSRC\_ASSN\_BY\_PROJROLE\_BY\_FP proj\_summ\_qty PROJWSRPT proj\_id WBSRSRC\_ASSN\_BY\_PROJRSRC\_BY\_FP proj\_id WBSRSRC\_ASSN\_BY\_PROJRSRC\_BY\_FP proj\_summ\_qty WBSRSRC\_ASSN\_SUMMARY\_BY\_FP proj\_id WBSRSRC\_ASSN\_SUMMARY\_BY\_FP proj\_summ\_qty SPIDMAP refrdel\_project\_bypass PREFER proj\_short\_len PREFER projinit\_admin\_user\_id PREFER task\_code\_len RELEASE proj\_id SUMPROJCOST proj\_id RELITEMS proj\_id SUMTASK proj\_id SUMTASKSPREAD proj\_id RISKCTRL proj\_id SUMTRSRC proj\_id TASK proj\_id TASK status\_code TASK task\_code RPT proj\_id RPTBATCH proj\_id TASKACTV actv\_code\_id TASKACTV actv\_code\_type\_id TASKACTV proj\_id RSRC employee\_code TASKDOC proj\_id TASKFDBK proj\_id PRMQUEUE status\_code TASKFIN proj\_id RSRCHOUR proj\_id RSRCHOUR status\_code TASKMEMO proj\_id TASKNOTE proj\_id TASKPRED pred\_proj\_id TASKPRED proj\_id TASKPROC proj\_id TASKRISK proj\_id TASKRSRC proj\_id PROJCOST proj\_id SCENPROJ proj\_id PROJECT intg\_proj\_type PROJECT orig\_proj\_id PROJECT proj\_id PROJECT proj\_short\_name PROJECT proj\_url PROJECT project\_flag PROJECT use\_project\_baseline\_flag PROJECT source\_proj\_id PROJECT sum\_base\_proj\_id PROJECT task\_code\_base PROJECT task\_code\_prefix PROJECT task\_code\_prefix\_flag PROJECT task\_code\_step TASKSUM proj\_id TASK proj\_id TASK status\_code TASK task\_code TASKSUMFIN proj\_id TASKACTV actv\_code\_id TASKACTV actv\_code\_type\_id TASKACTV proj\_id TASKUSER proj\_id TASKDOC proj\_id TASKWKSP proj\_id TASKFDBK proj\_id ACTVCODE actv\_code\_id ACTVCODE actv\_code\_name ACTVCODE actv\_code\_type\_id ACTVCODE parent\_actv\_code\_id TASKFIN proj\_id ACTVTYPE actv\_code\_type ACTVTYPE actv\_code\_type\_id ACTVTYPE actv\_code\_type\_scope ACTVTYPE proj\_id TASKMEMO proj\_id TASKNOTE proj\_id TASKPRED pred\_proj\_id TASKPRED proj\_id TASKPROC proj\_id TIMESHT status\_code TASKRISK proj\_id TASKRSRC proj\_id TMPLCATG project\_flag TASKSUM proj\_id TPROJMAP source\_proj\_id TPROJMAP target\_proj\_id BRE\_REGISTRY status\_code TASKSUMFIN proj\_id BUDGCHNG proj\_id BUDGCHNG status\_code TASKUSER proj\_id CALENDAR proj\_id TASKWKSP proj\_id TRSRCFIN proj\_id TRSRCSUM proj\_id TIMESHT status\_code TRSRCSUMFN proj\_id TMPLCATG project\_flag TSAUDIT proj\_id TSAUDIT proj\_short\_name TSAUDIT ts\_status\_code TRSRCFIN proj\_id TSDELEGATE proj\_id TRSRCSUM proj\_id TRSRCSUMFN proj\_id UACCESS proj\_id TSDELEGATE proj\_id UDFCODE parent\_udf\_code\_id UDFCODE udf\_code\_id UDFCODE udf\_code\_name UDFCODE parent\_udf\_code\_id UDFCODE udf\_code\_id UDFCODE udf\_code\_name UDFTYPE udf\_code\_short\_len UDFTYPE udf\_code\_short\_len DLTACTV actv\_code\_id DLTACTV parent\_actv\_code\_id UDFVALUE proj\_id UDFVALUE udf\_code\_id PROJEST proj\_est\_id PROJEST proj\_id UDFVALUE proj\_id UDFVALUE udf\_code\_id PROJFUND proj\_fund\_id PROJFUND proj\_id PROJISSU base\_proj\_id PROJISSU proj\_id PROJISSU status\_code ACTVCODE actv\_code\_id ACTVCODE actv\_code\_name ACTVCODE actv\_code\_type\_id ACTVCODE parent\_actv\_code\_id PROJPCAT proj\_catg\_id PROJPCAT proj\_catg\_type\_id PROJPCAT proj\_id ACTVTYPE actv\_code\_type ACTVTYPE actv\_code\_type\_id ACTVTYPE actv\_code\_type\_scope ACTVTYPE proj\_id PROJPROP proj\_id PROJRISK proj\_id PROJRISK risk\_code PROJRISK status\_code DOCSTAT doc\_status\_code USERWKSP proj\_id DOCUMENT proj\_id BRE\_REGISTRY status\_code USESSION hard\_drive\_code BUDGCHNG proj\_id BUDGCHNG status\_code PROJSHAR proj\_id CALENDAR proj\_id PROJTHRS proj\_id PROJTHRS status\_code EXPPROJ exp\_proj\_name EXPPROJ proj\_id VIEWPROP proj\_id EXTAPP proj\_id PROJWBS proj\_id PROJWBS proj\_node\_flag PROJWBS status\_code WBSBUDG proj\_id USERWKSP proj\_id PROJWSRPT proj\_id WBSMEMO proj\_id WBSRSRC proj\_id USESSAUD hard\_drive\_code DOCSTAT doc\_status\_code FORMPROJ proj\_id WBSSTEP proj\_id USESSION hard\_drive\_code DOCUMENT proj\_id WKFLTMPL project\_flag EXPPROJ exp\_proj\_name EXPPROJ proj\_id EXTAPP proj\_id Projects ProjectID Projects ProjectName WORKFLOW existing\_project\_flag WORKFLOW proj\_id REFRDEL proj\_id PROJRSRC\_VIEW proj\_id WORKSPACE proj\_id VIEWPROP proj\_id PROJROLE\_VIEW proj\_id Activities ProjectID PROJRSRCROLE\_VIEW proj\_id RELEASE proj\_id PROJSHAR\_BL\_VIEW proj\_id RELITEMS proj\_id ISSUHIST proj\_id PROJRSRC\_PS\_VIEW proj\_id FORMPROJ proj\_id PROJROLE\_PS\_VIEW proj\_id WBSBUDG proj\_id RISKCTRL proj\_id PROJRSRCROLE\_PS\_VIEW proj\_id WBSMEMO proj\_id PFLOAD\_VIEW proj\_id WBSRSRC proj\_id PFLOAD\_IP\_VIEW proj\_id

**ChatGPT said:**

Alright, mate! Based on the **table scan**, I see that project-related classifications might be stored in PCATVAL**,**PCATTYPE**, and**PROJPCAT, which deal with **project categories**. This is where **Strategic Objective** could be stored.

Since **Strategic Objective** is a **Project Code**, it likely belongs to a **Project Category Table** like:

* PCATVAL → Stores project category values
* PCATTYPE → Defines project category types
* PROJPCAT → Links project categories to projects

**🔹 Step 1: Find Where "Strategic Objective" is Stored**

Let’s check if **"Strategic Objective"** exists in any of these project category tables.

**✅ Search for "Strategic Objective" in**PCATVAL**(Project Category Values)**

sql

Copy

SELECT \* FROM PCATVAL WHERE proj\_catg\_name LIKE '%Strategic Objective%';

👉 If **Strategic Objective appears here**, then project codes (including Workpack ID) are stored in PCATVAL.

**🔹 Step 2: Find How**PCATVAL**Links to**Projects

Since project categories are stored separately from PROJECT, we need to see how they link up.

**✅ Check How**PCATVAL**Connects to**PROJECT

sql

Copy

SELECT \* FROM PROJPCAT WHERE proj\_catg\_id IN ( SELECT proj\_catg\_id FROM PCATVAL WHERE proj\_catg\_name LIKE '%Strategic Objective%' );

👉 This will show which **projects** are linked to **Strategic Objectives**.

**🔹 Step 3: Extract Workpack IDs from the Same Table**

Since the **table storing Strategic Objective will also store Workpack ID**, we can pull both.

**✅ Retrieve Workpack ID & Strategic Objective Together**

sql

Copy

SELECT p.ProjectID, pv.proj\_catg\_name AS WorkpackID, pt.proj\_catg\_name AS StrategicObjective FROM PROJECT p JOIN PROJPCAT pc ON p.ProjectID = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id JOIN PCATVAL pt ON pc.proj\_catg\_id = pt.proj\_catg\_id WHERE pt.proj\_catg\_name LIKE '%Strategic Objective%' OR pv.proj\_catg\_name LIKE '%Workpack ID%';

👉 **This will list** all projects with **Strategic Objective and Workpack ID**.

**🔹 What You Need to Do Next**

✅ **Run Step 1 to confirm if "Strategic Objective" exists in**PCATVAL**.**  
✅ **Run Step 2 to check how it links to**PROJECT**.**  
✅ **Run Step 3 to extract Workpack IDs and Strategic Objectives together.**

Once we confirm the table, we can **permanently integrate Workpack IDs into our SQL model**! 🚀 Let me know what you find, mate! 🔧

**You said:**

ok, I have found the value we need, the table is PCATVAL, and the Project code is in parent\_prog\_catg\_id, search for EC515

**ChatGPT said:**

Brilliant, mate! 🎯 Since **Workpack IDs (like EC515)** are stored in PCATVAL.parent\_proj\_catg\_id, let’s extract them properly and link them to projects.

**🔹 Step 1: Confirm EC515 Exists in**PCATVAL

Let’s first check if **EC515 is stored correctly**.

**✅ Run This Query**

sql

Copy

SELECT \* FROM PCATVAL WHERE parent\_proj\_catg\_id LIKE '%EC515%';

👉 This confirms **EC515 is in the database**.

**🔹 Step 2: Link Workpack IDs to Their Projects**

Since project categories are stored separately, we need to **join them with actual projects**.

**✅ Run This Query to Retrieve Workpack ID per Project**

sql

Copy

SELECT p.ProjectID, p.ProjectName, pv.proj\_catg\_name AS WorkpackID FROM PROJECT p JOIN PROJPCAT pc ON p.ProjectID = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id WHERE pv.parent\_proj\_catg\_id LIKE '%EC515%';

👉 **This will list** all projects linked to **Workpack ID = EC515**.

**🔹 Step 3: Extract All Workpack IDs for Scenario Comparisons**

If we need to **compare multiple Workpack IDs (not just EC515)**, modify the query:

sql

Copy

SELECT p.ProjectID, p.ProjectName, pv.proj\_catg\_name AS WorkpackID FROM PROJECT p JOIN PROJPCAT pc ON p.ProjectID = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id WHERE pv.proj\_catg\_name LIKE '%Workpack%';

👉 **This will extract ALL Workpack IDs assigned to projects.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to confirm EC515 exists in**PCATVAL**.**  
✅ **Run Step 2 to extract Workpack IDs and link them to projects.**  
✅ **Run Step 3 if you want to compare multiple Workpack IDs.**

Once confirmed, we can **store Workpack IDs in SQL permanently for Power BI dashboards!** 🚀 Let me know how it goes, mate! 🔧

**You said:**

Msg 207, Level 16, State 1, Line 3 Invalid column name 'ProjectID'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'ProjectID'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'ProjectName'. Total execution time: 00:00:00.016 - step 2 didnt work.... and Step 1, just returned the headers of the table, not anything in it....which is really frustrating

**ChatGPT said:**

Ah, mate, I get the frustration! 😤 Let’s **fix this properly step by step** so we can pull Workpack IDs correctly.

Since Step 1 **didn’t return results**, it means either:  
1️⃣ **EC515 doesn’t exist in**PCATVAL.parent\_proj\_catg\_id 🤔  
2️⃣ **We’re searching in the wrong format (e.g., extra spaces, case sensitivity, etc.)**

**🔹 Step 1: Double-Check If EC515 Exists in**PCATVAL

Let’s do a **broader search** to make sure the value is there.

**✅ Run This Query to See What’s in**PCATVAL

sql

Copy

SELECT TOP 20 \* FROM PCATVAL;

👉 **Manually check the column names and confirm where Workpack IDs are stored.**

**🔹 Step 2: Search for Any Workpack IDs in**PCATVAL

If **Step 1 returns data**, but EC515 is missing, let’s check if any Workpack IDs exist at all.

sql

Copy

SELECT DISTINCT parent\_proj\_catg\_id FROM PCATVAL;

👉 **This will list unique values** from parent\_proj\_catg\_id.  
👉 Look for **EC515, DHBL, or any other Workpack IDs**.

**🔹 Step 3: Search for EC515 in the Right Column**

If **Step 2 shows EC515 exists**, but Step 1 didn’t return data, try searching **without wildcards**:

sql

Copy

SELECT \* FROM PCATVAL WHERE parent\_proj\_catg\_id = 'EC515';

👉 **This ensures we’re searching for an exact match.**

**🔹 Step 4: Fix the**ProjectID**Error in Step 2**

The error **"Invalid column name 'ProjectID'"** means PROJECT might use a different column name.

**✅ Find the Correct Column Name in**PROJECT

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT';

👉 **Look for what the correct project ID column is called (e.g.,**proj\_id**instead of**ProjectID**).**

**🔹 Step 5: Re-Run the Query with the Correct Column Name**

Once we confirm the **correct project ID column**, update Step 2.

sql

Copy

SELECT p.proj\_id, p.proj\_short\_name, pv.proj\_catg\_name AS WorkpackID FROM PROJECT p JOIN PROJPCAT pc ON p.proj\_id = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id WHERE pv.parent\_proj\_catg\_id = 'EC515';

👉 **Make sure**proj\_id**and**proj\_short\_name**are the correct column names from Step 4.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check what’s in**PCATVAL**.**  
✅ **Run Step 2 to see if Workpack IDs exist.**  
✅ **Run Step 3 to confirm EC515 is stored correctly.**  
✅ **Run Step 4 to find the correct project column names.**  
✅ **Run Step 5 with the correct column names.**

This will **guarantee** we pull Workpack IDs correctly and fix any errors! 🚀 Let me know what Step 1 & Step 4 return, and we’ll move forward! 🔧

**You said:**

parent\_proj\_catg\_id is not in the headers list

**ChatGPT said:**

Ah, mate, that explains everything! **If**parent\_proj\_catg\_id**isn’t in the headers of**PCATVAL, then we’ve been searching for a column that **doesn’t exist**.

No worries—we’ll **fix this by properly identifying where Workpack IDs are stored**.

**🔹 Step 1: Find the Actual Column That Stores Workpack IDs**

Since parent\_proj\_catg\_id**doesn’t exist**, we need to look for a column that **contains project categories**.

**✅ Run This Query to List All Columns in**PCATVAL

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PCATVAL';

👉 **Look for anything related to**proj\_catg**,**proj\_code**,**parent**, or**value**.**

**🔹 Step 2: Look for Workpack IDs in**PCATVAL

Once we find a **relevant column name**, let’s search for **EC515** or any Workpack IDs.

**✅ Replace**[CorrectColumn]**with the Column Found in Step 1**

sql

Copy

SELECT \* FROM PCATVAL WHERE [CorrectColumn] LIKE '%EC515%';

👉 **If EC515 is found, we’ve got the right column!**

**🔹 Step 3: Link Workpack IDs to Projects**

Once we confirm **where Workpack IDs are stored**, we’ll link them to PROJECT.

**✅ Find the Correct Column Name in**PROJECT

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT';

👉 **Check what the actual project ID column is called (e.g.,**proj\_id**instead of**ProjectID**).**

**🔹 Step 4: Retrieve Workpack IDs with the Correct Column Name**

Once we know the correct column names, **update the query to extract Workpack IDs**.

sql

Copy

SELECT p.proj\_id, p.proj\_short\_name, pv.[CorrectColumn] AS WorkpackID FROM PROJECT p JOIN PROJPCAT pc ON p.proj\_id = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id WHERE pv.[CorrectColumn] LIKE '%EC515%';

👉 **Replace**[CorrectColumn]**with the actual Workpack ID column from**PCATVAL**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find the correct column in**PCATVAL**.**  
✅ **Run Step 2 to check if EC515 is stored in that column.**  
✅ **Run Step 3 to find the correct project ID column in**PROJECT**.**  
✅ **Run Step 4 to link Workpack IDs to Projects.**

This will **guarantee** we extract Workpack IDs properly! 🚀 Let me know what Step 1 finds, and we’ll move forward! 🔧

**You said:**

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PCATVAL';

**ChatGPT said:**

Great! **Run that query**, and let me know the **list of column names** that appear in PCATVAL.

👉 Once we see the actual column names, we’ll know **where Workpack IDs (like EC515) are stored** and adjust our queries accordingly. 🚀

**You said:**

proj\_catg\_id proj\_catg\_type\_id seq\_num proj\_catg\_short\_name parent\_proj\_catg\_id proj\_catg\_name proj\_catg\_wt update\_date update\_user create\_date create\_user delete\_session\_id delete\_date proj\_catg\_id proj\_catg\_type\_id seq\_num proj\_catg\_short\_name parent\_proj\_catg\_id proj\_catg\_name proj\_catg\_wt update\_date update\_user create\_date create\_user delete\_session\_id delete\_date

**ChatGPT said:**

**Boom!** 🎯 Now we’ve got the actual columns in PCATVAL!

The key columns we care about:  
✔ proj\_catg\_name → Likely stores the **category name** (could be "Workpack ID")  
✔ parent\_proj\_catg\_id → Might define a parent-child relationship between categories  
✔ proj\_catg\_short\_name → Might hold shorthand codes (like EC515)

**🔹 Step 1: Search for "Workpack ID" in**proj\_catg\_name

Let’s check if **"Workpack ID" is stored as a category** in proj\_catg\_name.

**✅ Run This Query**

sql

Copy

SELECT \* FROM PCATVAL WHERE proj\_catg\_name LIKE '%Workpack%';

👉 If "Workpack ID" appears here, that’s **our key field**!

**🔹 Step 2: Search for EC515 in Other Columns**

Since we’re looking for **EC515**, let’s check if it appears in proj\_catg\_short\_name or parent\_proj\_catg\_id.

**✅ Try This Query**

sql

Copy

SELECT \* FROM PCATVAL WHERE proj\_catg\_short\_name LIKE '%EC515%' OR parent\_proj\_catg\_id LIKE '%EC515%';

👉 If EC515 is here, that confirms **where Workpack IDs are stored**.

**🔹 Step 3: Link Workpack IDs to Projects**

Once we confirm where EC515 is stored, we’ll **link it to projects**.

**✅ Find the Correct Column Name in**PROJECT

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT';

👉 **Look for**proj\_id**or similar.**

**🔹 Step 4: Extract Workpack IDs with the Correct Column Names**

Once we know where Workpack IDs are stored, we’ll **join them to projects**.

sql

Copy

SELECT p.proj\_id, p.proj\_short\_name, pv.proj\_catg\_name AS WorkpackID FROM PROJECT p JOIN PROJPCAT pc ON p.proj\_id = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id WHERE pv.proj\_catg\_name LIKE '%Workpack ID%';

👉 **Adjust column names based on what Step 1 & 2 return.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check if "Workpack ID" exists in**proj\_catg\_name**.**  
✅ **Run Step 2 to confirm if EC515 exists in**proj\_catg\_short\_name**or**parent\_proj\_catg\_id**.**  
✅ **Run Step 3 to find the correct project column names.**  
✅ **Run Step 4 to extract Workpack IDs linked to projects.**

Once confirmed, we’ll **store Workpack IDs permanently for Power BI reporting!** 🚀 Let me know what Step 1 & Step 2 return! 🔧

**You said:**

step 1, didnt return anything....just a bunch of headers.

**ChatGPT said:**

Alright, mate! Since **Step 1 didn’t return anything**, that means **"Workpack ID" isn't explicitly stored in**proj\_catg\_name. No worries—we’ll find it another way.

Let’s **change strategy** and directly look for any **actual Workpack IDs (like EC515)** in the data.

**🔹 Step 1: Find Any Workpack IDs in**PCATVAL

Since **"Workpack ID" isn’t stored as a category**, we need to search for **EC515 or similar values**.

**✅ Try This Query**

sql

Copy

SELECT DISTINCT proj\_catg\_short\_name, proj\_catg\_name, parent\_proj\_catg\_id FROM PCATVAL;

👉 **Look through the results and check if EC515 appears in any column.**

**🔹 Step 2: Check If EC515 Is Stored in**parent\_proj\_catg\_id

If Step 1 **returns EC515 in**parent\_proj\_catg\_id, we need to find what project categories it belongs to.

**✅ Try This Query**

sql

Copy

SELECT \* FROM PCATVAL WHERE parent\_proj\_catg\_id IS NOT NULL;

👉 **This helps confirm if Workpack IDs are stored as parent categories.**

**🔹 Step 3: Link Workpack IDs to Projects**

Once we confirm **where EC515 is stored**, we’ll join it to projects.

**✅ Run This Query (Once We Confirm the Right Column)**

sql

Copy

SELECT p.proj\_id, p.proj\_short\_name, pv.proj\_catg\_short\_name AS WorkpackID FROM PROJECT p JOIN PROJPCAT pc ON p.proj\_id = pc.proj\_id JOIN PCATVAL pv ON pc.proj\_catg\_id = pv.proj\_catg\_id WHERE pv.parent\_proj\_catg\_id IS NOT NULL;

👉 **This will list all projects linked to Workpack IDs.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find EC515 in any column.**  
✅ **Run Step 2 to check if Workpack IDs are stored as parent categories.**  
✅ **Tell me where EC515 appears, and I’ll adjust Step 3 to extract the Workpack IDs properly!** 🚀

Let me know what you find, mate! 🔧

**You said:**

ok, scratch this, its not working, and I cant spend anymore time on it. Can we go back to the Projects table please bud

**ChatGPT said:**

Got it, mate! No stress—we’ll **drop the Workpack ID hunt for now** and shift focus back to the **Projects table**.

**🔹 What’s Next for the**Projects**Table?**

Are we:  
✅ **Creating a new**Projects**table from scratch?**  
✅ **Modifying an existing**Projects**table (adding/removing columns)?**  
✅ **Populating it with new data from P6?**  
✅ **Linking it to other tables (Activities, Resources, etc.)?**

Let me know what you want to focus on, and I’ll walk you through it step by step! 🚀🔧

**2/2**

**You said:**

we have already created the table in the dbo structure, its dbo.Projects. we just need to populate data from P6 into it now, on the columns we specified when the table was set up ok.

**ChatGPT said:**

Perfect, mate! Since dbo.Projects is already created, we just need to **pull the relevant project data from P6 and insert it into SQL**.

**🔹 Step 1: Check the**dbo.Projects**Table Structure**

Before inserting data, let’s **confirm the column names** in dbo.Projects to match them with P6.

**✅ Run This Query to List the Columns in**dbo.Projects

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Projects' AND TABLE\_SCHEMA = 'dbo';

👉 This ensures **we insert data into the correct columns**.

**🔹 Step 2: Check Where Project Data is Stored in P6**

Now, let’s **find the correct P6 table** that holds the project data.

**✅ Run This Query to List Columns in P6’s**PROJECT**Table**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT';

👉 This helps us **match columns between**PROJECT**in P6 and**dbo.Projects**in SQL.**

**🔹 Step 3: Insert Project Data from P6 into**dbo.Projects

Once we know **which columns match**, let’s **insert the project data**.

**✅ Run This Query to Populate**dbo.Projects

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **This copies all project records from P6 into**dbo.Projects**.**

**🔹 Step 4: Verify Data is Inserted Correctly**

After running the insert query, let’s **check if data is properly stored** in dbo.Projects.

**✅ Run This Query to View the First 10 Projects**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 If **projects appear correctly**, 🎉 **we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check**dbo.Projects**structure.**  
✅ **Run Step 2 to check P6’s**PROJECT**table structure.**  
✅ **Run Step 3 to insert project data.**  
✅ **Run Step 4 to verify the data.**

Once this is done, we can **start linking projects to activities, resources, and scheduling data!** 🚀 Let me know how it goes! 🔧

**You said:**

ProjectID ProjectName StartDate EndDate - can move on to step 2 now.

**ChatGPT said:**

Perfect, mate! **Step 1 confirmed** the columns in dbo.Projects:  
✔ **ProjectID**  
✔ **ProjectName**  
✔ **StartDate**  
✔ **EndDate**

Now, let’s **move to Step 2** and check where P6 stores this data.

**🔹 Step 2: Check P6’s**PROJECT**Table Structure**

Let’s confirm the **correct columns** in P6’s PROJECT table.

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'PROJECT';

👉 This will **list all columns** in the PROJECT table.  
👉 We need to **match** them with dbo.Projects.

**🔹 Step 3: Identify Matching Columns in**PROJECT

P6 **likely** uses the following:  
✔ proj\_id → **Matches**ProjectID  
✔ proj\_short\_name → **Matches**ProjectName  
✔ plan\_start\_date → **Matches**StartDate  
✔ plan\_end\_date → **Matches**EndDate

Once **Step 2 confirms these columns exist**, we can move to **Step 3: Insert Data**. 🚀

Let me know what **Step 2 returns**, mate! 🔧

**You said:**

proj\_id fy\_start\_month\_num chng\_eff\_cmp\_pct\_flag rsrc\_self\_add\_flag allow\_complete\_flag rsrc\_multi\_assign\_flag checkout\_flag project\_flag step\_complete\_flag cost\_qty\_recalc\_flag sum\_only\_flag batch\_sum\_flag name\_sep\_char def\_complete\_pct\_type proj\_short\_name acct\_id orig\_proj\_id source\_proj\_id base\_type\_id clndr\_id sum\_base\_proj\_id task\_code\_base task\_code\_step priority\_num wbs\_max\_sum\_level risk\_level strgy\_priority\_num last\_checksum critical\_drtn\_hr\_cnt def\_cost\_per\_qty last\_recalc\_date plan\_start\_date plan\_end\_date scd\_end\_date add\_date sum\_data\_date last\_tasksum\_date fcst\_start\_date def\_duration\_type task\_code\_prefix guid def\_qty\_type add\_by\_name web\_local\_root\_path proj\_url def\_rate\_type act\_this\_per\_link\_flag def\_task\_type act\_pct\_link\_flag add\_act\_remain\_flag critical\_path\_type task\_code\_prefix\_flag def\_rollup\_dates\_flag rem\_target\_link\_flag reset\_planned\_flag allow\_neg\_act\_flag rsrc\_id msp\_managed\_flag msp\_update\_actuals\_flag checkout\_date checkout\_user\_id sum\_assign\_level last\_fin\_dates\_id use\_project\_baseline\_flag last\_baseline\_update\_date ts\_rsrc\_vw\_compl\_asgn\_flag ts\_rsrc\_mark\_act\_finish\_flag ts\_rsrc\_vw\_inact\_actv\_flag cr\_external\_key apply\_actuals\_date description intg\_proj\_type matrix\_id location\_id control\_updates\_flag hist\_interval hist\_level update\_date update\_user create\_date create\_user delete\_session\_id delete\_date proj\_id fy\_start\_month\_num chng\_eff\_cmp\_pct\_flag rsrc\_self\_add\_flag allow\_complete\_flag rsrc\_multi\_assign\_flag checkout\_flag project\_flag step\_complete\_flag cost\_qty\_recalc\_flag sum\_only\_flag batch\_sum\_flag name\_sep\_char def\_complete\_pct\_type proj\_short\_name acct\_id orig\_proj\_id source\_proj\_id base\_type\_id clndr\_id sum\_base\_proj\_id task\_code\_base task\_code\_step priority\_num wbs\_max\_sum\_level risk\_level strgy\_priority\_num last\_checksum critical\_drtn\_hr\_cnt def\_cost\_per\_qty last\_recalc\_date plan\_start\_date plan\_end\_date scd\_end\_date add\_date sum\_data\_date last\_tasksum\_date fcst\_start\_date def\_duration\_type task\_code\_prefix guid def\_qty\_type add\_by\_name web\_local\_root\_path proj\_url def\_rate\_type act\_this\_per\_link\_flag def\_task\_type act\_pct\_link\_flag add\_act\_remain\_flag critical\_path\_type task\_code\_prefix\_flag def\_rollup\_dates\_flag rem\_target\_link\_flag reset\_planned\_flag allow\_neg\_act\_flag rsrc\_id msp\_managed\_flag msp\_update\_actuals\_flag checkout\_date checkout\_user\_id sum\_assign\_level last\_fin\_dates\_id use\_project\_baseline\_flag last\_baseline\_update\_date ts\_rsrc\_vw\_compl\_asgn\_flag ts\_rsrc\_mark\_act\_finish\_flag ts\_rsrc\_vw\_inact\_actv\_flag cr\_external\_key apply\_actuals\_date description intg\_proj\_type matrix\_id location\_id control\_updates\_flag hist\_interval hist\_level update\_date update\_user create\_date create\_user delete\_session\_id delete\_date - Step 2 is complete, these are the headers.

**ChatGPT said:**

**Nice one, mate!** 🎯 Now that we have the **P6**PROJECT**table headers**, we can map them to dbo.Projects and **start inserting data**.

**🔹 Step 3: Match P6 Columns to**dbo.Projects

We need to **map** P6’s PROJECT table to dbo.Projects:

| dbo.Projects**Column** | **P6**PROJECT**Column** |
| --- | --- |
| ProjectID | proj\_id |
| ProjectName | proj\_short\_name |
| StartDate | plan\_start\_date |
| EndDate | plan\_end\_date |

✅ **These match perfectly**, so we’re ready to insert the data.

**🔹 Step 4: Insert Data from P6 into**dbo.Projects

Now, let’s **populate**dbo.Projects using the correct columns from PROJECT.

**✅ Run This Query**

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

👉 **This will insert all project records from P6 into SQL Server.**

**🔹 Step 5: Verify the Inserted Data**

After running the **INSERT** query, let’s **check if data was successfully stored**.

**✅ Run This Query**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If projects appear, 🎉 we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 4 to insert project data into**dbo.Projects**.**  
✅ **Run Step 5 to verify that projects were inserted correctly.**  
✅ **Let me know if you get any errors or if everything works!** 🚀

Once this is done, we can move on to **linking projects to activities, resources, and schedules!** 🔧

**You said:**

Started executing query at Line 1 Msg 544, Level 16, State 1, Line 1 Cannot insert explicit value for identity column in table 'Projects' when IDENTITY\_INSERT is set to OFF. Total execution time: 00:00:00.015 on step 4 bud

**ChatGPT said:**

Ah, mate, that’s an **easy fix!** ⚡ The error means ProjectID**is an**IDENTITY**column**, and SQL Server **won’t allow inserting explicit values** unless we enable IDENTITY\_INSERT.

**🔹 Step 4 (Fixed) – Enable**IDENTITY\_INSERT**& Insert Data**

Since ProjectID is auto-generated in dbo.Projects, we need to **temporarily allow inserting values into it**.

**✅ Run This Query**

sql

Copy

SET IDENTITY\_INSERT dbo.Projects ON; INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT; SET IDENTITY\_INSERT dbo.Projects OFF;

👉 **This enables inserting into**ProjectID**, inserts the data, then turns it off again.**

**🔹 Step 5: Verify the Data**

Once inserted, let’s **check if projects are now in**dbo.Projects.

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If data appears, 🎉 problem solved!**

**🔹 What You Need to Do Next**

✅ **Run the fixed Step 4 query.**  
✅ **Run Step 5 to verify the insert.**  
✅ **Let me know if you get any errors!** 🚀

Once confirmed, we’ll move on to **linking projects with activities and resources.** 🔧🔥

**You said:**

we need to keep the Project ID, from P6 and reference it to the other tables we have, so we cant auto generate project ids, does that clarify things for you. Project ID might be used as the primary key on the tables on some instances.

**ChatGPT said:**

**Crystal clear, mate!** ✅ Since we **need to retain the**ProjectID**from P6** and use it as a primary key across multiple tables, we must **remove the auto-generated**IDENTITY(1,1)**setting from**ProjectID in dbo.Projects.

**🔹 Step 1: Check If**ProjectID**Has**IDENTITY**Set**

Before we can fix this, let’s confirm if ProjectID**has**IDENTITY(1,1)**enabled**.

**✅ Run This Query**

sql

Copy

SELECT name, is\_identity FROM sys.columns WHERE object\_id = OBJECT\_ID('dbo.Projects') AND name = 'ProjectID';

👉 **If**is\_identity = 1, it means ProjectID is set as an IDENTITY column.

**🔹 Step 2: Remove the**IDENTITY**Property from**ProjectID

Since we **can’t directly remove**IDENTITY, we have to:  
1️⃣ **Create a new table without**IDENTITY  
2️⃣ **Copy data over**  
3️⃣ **Drop the old table & rename the new one**

**✅ Run These Queries (Fix for**IDENTITY**Issue)**

sql

Copy

*-- 1️⃣ Create a new table without IDENTITY* CREATE TABLE dbo.Projects\_New ( ProjectID INT PRIMARY KEY, *-- No IDENTITY here* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL ); *-- 2️⃣ Copy any existing data (if applicable)* INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; *-- 3️⃣ Drop the old table* DROP TABLE dbo.Projects; *-- 4️⃣ Rename the new table to dbo.Projects* EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

👉 **Now**ProjectID**is no longer auto-generated**, and we can insert P6 data directly.

**🔹 Step 3: Insert P6 Data into**dbo.Projects

Once we fix the table, we can **insert data from P6 without issues**.

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **Now, we retain the exact P6**ProjectID**.**

**🔹 Step 4: Verify Data**

After inserting, **check if everything looks good**:

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If projects appear correctly, 🎉 we’re all set!**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to confirm if**ProjectID**has**IDENTITY**.**  
✅ **If**IDENTITY = 1**, run Step 2 to fix the table.**  
✅ **Run Step 3 to insert data from P6.**  
✅ **Run Step 4 to verify everything works.**

Once done, we can **move to linking projects with activities and resources.** 🚀 Let me know how it goes, mate! 🔧🔥

**You said:**

ProjectID 1 - thats step 1 complete, now step 2.

**ChatGPT said:**

**Nice one, mate!** ✅ Since **Step 1 confirmed**ProjectID**is an**IDENTITY**column**, we now need to **remove**IDENTITY and keep ProjectID as a fixed primary key.

**🔹 Step 2: Remove**IDENTITY**from**ProjectID

Since we **can’t directly remove**IDENTITY(1,1), we need to:  
1️⃣ **Create a new table (**Projects\_New**) without**IDENTITY  
2️⃣ **Copy data over**  
3️⃣ **Drop the old table**  
4️⃣ **Rename the new table back to**Projects

**✅ Run These Queries**

sql

Copy

*-- 1️⃣ Create a new table without IDENTITY* CREATE TABLE dbo.Projects\_New ( ProjectID INT PRIMARY KEY, *-- No IDENTITY here* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL ); *-- 2️⃣ Copy existing data into the new table* SET IDENTITY\_INSERT dbo.Projects ON; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; SET IDENTITY\_INSERT dbo.Projects OFF; *-- 3️⃣ Drop the old table* DROP TABLE dbo.Projects; *-- 4️⃣ Rename the new table to dbo.Projects* EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

✅ **Now,**ProjectID**is no longer auto-generated, and we can insert P6 data freely.**

**🔹 Step 3: Insert P6 Data into**dbo.Projects

Now that ProjectID is no longer IDENTITY, we can insert **data from P6** without issues.

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **This will insert P6 project data while keeping the original**ProjectID**.**

**🔹 Step 4: Verify Data**

After inserting, let’s **check if the data is stored correctly**:

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If the projects appear correctly, 🎉 we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 2 to remove**IDENTITY**from**ProjectID**.**  
✅ **Run Step 3 to insert P6 project data.**  
✅ **Run Step 4 to verify the data.**

Once done, we can move to **linking projects with activities, resources, and schedules.** 🚀 Let me know how it goes, mate! 🔧🔥

**You said:**

Msg 3726, Level 16, State 1, Line 18 Could not drop object 'dbo.Projects' because it is referenced by a FOREIGN KEY constraint. Msg 15335, Level 11, State 1, Procedure sp\_rename, Line 576 Error: The new name 'Projects' is already in use as a object name and would cause a duplicate that is not permitted. Total execution time: 00:00:00.198, - Step 2 resulted in an error.

**ChatGPT said:**

**Got it, mate!** ✅ The error is caused by **foreign key constraints** in other tables that reference dbo.Projects. Since SQL won’t let us **drop**Projects**while it’s being referenced**, we’ll handle this differently.

**🔹 Step 2 (Fixed) – Remove**IDENTITY**Without Dropping the Table**

Since we **can’t drop**dbo.Projects, we will:  
1️⃣ **Temporarily disable foreign key constraints**  
2️⃣ **Create a new column (**ProjectID\_New**) without**IDENTITY  
3️⃣ **Copy data into**ProjectID\_New  
4️⃣ **Remove the old**ProjectID**column**  
5️⃣ **Rename**ProjectID\_New**back to**ProjectID  
6️⃣ **Re-enable foreign keys**

**✅ Step 2 (Fixed) – Modify**ProjectID**Without Dropping the Table**

sql

Copy

*-- 1️⃣ Temporarily disable foreign key constraints* ALTER TABLE [dbo.OtherTable] NOCHECK CONSTRAINT ALL; *-- 2️⃣ Add a new ProjectID column (without IDENTITY)* ALTER TABLE dbo.Projects ADD ProjectID\_New INT NULL; *-- 3️⃣ Copy existing data into the new column* UPDATE dbo.Projects SET ProjectID\_New = ProjectID; *-- 4️⃣ Drop the old ProjectID column* ALTER TABLE dbo.Projects DROP COLUMN ProjectID; *-- 5️⃣ Rename the new column to ProjectID* EXEC sp\_rename 'dbo.Projects.ProjectID\_New', 'ProjectID', 'COLUMN'; *-- 6️⃣ Re-enable foreign key constraints* ALTER TABLE [dbo.OtherTable] CHECK CONSTRAINT ALL;

👉 **Replace**[dbo.OtherTable]**with the actual table(s) that reference**Projects (use sys.foreign\_keys to find them).

✅ **Now,**ProjectID**is no longer**IDENTITY**, and all foreign key links remain intact.**

**🔹 Step 3: Insert P6 Data into**dbo.Projects

Now that ProjectID is fixed, we can **insert P6 data** without issues.

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **This ensures P6 project data is inserted correctly.**

**🔹 Step 4: Verify Data**

After inserting, let’s **check if the data is stored correctly**:

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If the projects appear correctly, 🎉 we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 2 (Fixed) to remove**IDENTITY**without dropping**Projects**.**  
✅ **Run Step 3 to insert P6 project data.**  
✅ **Run Step 4 to verify the data.**

Once done, we can move on to **linking projects with activities, resources, and schedules!** 🚀 Let me know how it goes, mate! 🔧🔥

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 9 Invalid column name 'ProjectID\_New'. Total execution time: 00:00:00.004 another error mate, we not winning here are we.

**ChatGPT said:**

Yeah, mate, this one’s putting up a fight! 🥊 But don’t worry, we’ll **get it sorted step by step**.

The error **"Invalid column name 'ProjectID\_New'"** means the column **wasn't created** before the update. Let’s take a **slightly different approach** to work around SQL Server's IDENTITY limitation.

**🔹 Step 1: Find Out If**ProjectID**Has**IDENTITY**Again**

Before making changes, let's confirm whether SQL still considers ProjectID as an identity column.

**✅ Run This Query**

sql

Copy

SELECT name, is\_identity FROM sys.columns WHERE object\_id = OBJECT\_ID('dbo.Projects') AND name = 'ProjectID';

👉 **If**is\_identity = 1, we know we still need to remove it.

**🔹 Step 2: Workaround to Remove**IDENTITY**Without Dropping Data**

Since SQL doesn’t let us **directly remove**IDENTITY, we’ll **rebuild the table correctly**:

**✅ Try This Fix**

sql

Copy

*-- 1️⃣ Create a new table without IDENTITY* CREATE TABLE dbo.Projects\_New ( ProjectID INT PRIMARY KEY, *-- No IDENTITY here* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL ); *-- 2️⃣ Copy data over* INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; *-- 3️⃣ Drop foreign key constraints before dropping the table (Find these using sys.foreign\_keys if needed)* ALTER TABLE [dbo.OtherTable] DROP CONSTRAINT [FK\_YourForeignKeyName]; *-- 4️⃣ Drop the old table* DROP TABLE dbo.Projects; *-- 5️⃣ Rename the new table* EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

👉 **If you have foreign keys, we’ll need to drop and re-add them (Step 3).**  
👉 If Step 3 fails, let me know and I’ll help you find the correct FK names.

**🔹 Step 3: Insert P6 Data into**dbo.Projects

Once ProjectID is **no longer an identity column**, we can **insert P6 data freely**.

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **This ensures P6 project data is inserted correctly.**

**🔹 Step 4: Verify Everything**

Now, **check if data was inserted properly**:

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If projects appear correctly, 🎉 we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check if**ProjectID**still has**IDENTITY**.**  
✅ **Run Step 2 (Fixed) to remove**IDENTITY**by rebuilding the table.**  
✅ **Run Step 3 to insert P6 data.**  
✅ **Run Step 4 to verify the data.**

If SQL **blocks Step 3 because of foreign keys**, let me know and we’ll **find the FKs and remove them correctly**. 🚀 Let’s get this sorted! 🔧🔥

**You said:**

ProjectID 1 - this is the output of step 1

**ChatGPT said:**

**Alright, mate!** ✅ Since **Step 1 confirms**ProjectID**still has**IDENTITY, we’ll now **remove**IDENTITY**properly** without losing data.

**🔹 Step 2: Remove**IDENTITY**Without Losing Data**

Since **we can’t directly remove**IDENTITY(1,1), we need to:

1️⃣ **Create a new table (**Projects\_New**) without**IDENTITY  
2️⃣ **Copy all existing data**  
3️⃣ **Drop foreign key constraints (if needed)**  
4️⃣ **Drop the old table (**Projects**)**  
5️⃣ **Rename**Projects\_New**to**Projects

**✅ Step 2 (Fixed) – Recreate**Projects**Without**IDENTITY

sql

Copy

*-- 1️⃣ Create a new table without IDENTITY* CREATE TABLE dbo.Projects\_New ( ProjectID INT PRIMARY KEY, *-- No IDENTITY here* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL ); *-- 2️⃣ Copy existing data into the new table* SET IDENTITY\_INSERT dbo.Projects ON; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; SET IDENTITY\_INSERT dbo.Projects OFF;

👉 **This ensures all data is copied correctly before we drop the old table.**

**✅ Step 3: Drop Foreign Keys Before Dropping**Projects**(If Needed)**

If Projects is **linked to other tables**, we need to **drop constraints first**.

**Find Foreign Keys (If Step 4 Fails)**

sql

Copy

SELECT name FROM sys.foreign\_keys WHERE referenced\_object\_id = OBJECT\_ID('dbo.Projects');

👉 If this **returns any FK names**, we need to **drop them before Step 4** using:

sql

Copy

ALTER TABLE [dbo.OtherTable] DROP CONSTRAINT [FK\_YourForeignKeyName];

Replace dbo.OtherTable with the actual table name.

**✅ Step 4: Drop the Old Table & Rename the New One**

sql

Copy

DROP TABLE dbo.Projects; EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

👉 **Now,**Projects**no longer has**IDENTITY**! 🎉**

**🔹 Step 5: Insert P6 Data into**dbo.Projects

Now that ProjectID is **a normal primary key**, let’s **insert data from P6**:

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **This ensures P6 project data is inserted correctly.**

**🔹 Step 6: Verify Everything**

Let’s **check if projects are stored correctly**:

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If projects appear correctly, 🎉 we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 2 (Fixed) to create**Projects\_New**and copy data.**  
✅ **Run Step 3 if Step 4 fails due to foreign keys.**  
✅ **Run Step 4 to drop the old table and rename the new one.**  
✅ **Run Step 5 to insert P6 data.**  
✅ **Run Step 6 to verify the data.**

🚀 Let’s get this sorted, mate! Let me know if any issues pop up! 🔧🔥

**You said:**

Started executing query at Line 1 Msg 2714, Level 16, State 6, Line 2 There is already an object named 'Projects\_New' in the database. Total execution time: 00:00:00.017 - step 2, created this error....do I need to run these pieces of code individually?

**ChatGPT said:**

Yep, mate! Since Projects\_New**already exists**, we need to **drop it first** before creating a fresh version.

You can **run the queries individually** to avoid any errors, but here’s the **fixed full process** that should work in one go.

**🔹 Step 1: Drop**Projects\_New**If It Already Exists**

Since SQL **won’t create a table that already exists**, we **drop it first**.

**✅ Run This Query First**

sql

Copy

IF OBJECT\_ID('dbo.Projects\_New', 'U') IS NOT NULL DROP TABLE dbo.Projects\_New;

👉 **This deletes**Projects\_New**only if it already exists.**

**🔹 Step 2: Recreate**Projects**Without**IDENTITY

Now, we **create**Projects\_New**without**IDENTITY and **copy data over**.

**✅ Run This Query**

sql

Copy

*-- 1️⃣ Create a new table without IDENTITY* CREATE TABLE dbo.Projects\_New ( ProjectID INT PRIMARY KEY, *-- No IDENTITY here* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL ); *-- 2️⃣ Copy existing data into the new table* SET IDENTITY\_INSERT dbo.Projects ON; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; SET IDENTITY\_INSERT dbo.Projects OFF;

✅ **Now, we have a clean table without**IDENTITY**.**

**🔹 Step 3: Drop the Old**Projects**Table**

Since Projects is **linked to other tables**, we may need to **drop foreign key constraints first**.

**Find Foreign Keys Before Dropping**Projects

sql

Copy

SELECT name FROM sys.foreign\_keys WHERE referenced\_object\_id = OBJECT\_ID('dbo.Projects');

👉 If this **returns FK names**, we **drop them first**:

sql

Copy

ALTER TABLE [dbo.OtherTable] DROP CONSTRAINT [FK\_YourForeignKeyName];

(Replace dbo.OtherTable with the actual table name.)

**Now, Drop the Old**Projects**Table**

sql

Copy

DROP TABLE dbo.Projects;

**🔹 Step 4: Rename**Projects\_New**to**Projects

sql

Copy

EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

✅ **Now,**Projects**is clean and ready for P6 data.**

**🔹 Step 5: Insert P6 Data into**dbo.Projects

Now that ProjectID is **a normal primary key**, let’s **insert data from P6**.

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

✅ **This ensures P6 project data is inserted correctly.**

**🔹 Step 6: Verify Everything**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If projects appear correctly, 🎉 we’re done!**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to drop**Projects\_New**if it exists.**  
✅ **Run Step 2 to create**Projects\_New**and copy data.**  
✅ **Run Step 3 if Step 4 fails due to foreign keys.**  
✅ **Run Step 4 to rename**Projects\_New**to**Projects**.**  
✅ **Run Step 5 to insert P6 data.**  
✅ **Run Step 6 to verify the data.**

🚀 Let’s get this sorted, mate! Let me know if any issues pop up! 🔧🔥

**You said:**

on step 2...do you want me to run the code individually or all of that in one query?

**ChatGPT said:**

Good question, mate! **To avoid errors**, it’s best to **run the queries individually, step by step** rather than all at once.

**🔹 Step-by-Step Execution Plan**

**✅ Step 1: Drop**Projects\_New**If It Exists**

Run this **first** to clear out any old data.

sql

Copy

IF OBJECT\_ID('dbo.Projects\_New', 'U') IS NOT NULL DROP TABLE dbo.Projects\_New;

👉 **Wait for confirmation that it executed successfully before moving on.**

**✅ Step 2: Create a New**Projects**Table (Without**IDENTITY**)**

Run this **second** to create a clean table.

sql

Copy

CREATE TABLE dbo.Projects\_New ( ProjectID INT PRIMARY KEY, *-- No IDENTITY here* ProjectName NVARCHAR(255) NOT NULL, StartDate DATE NULL, EndDate DATE NULL );

👉 **Once created successfully, move to Step 3.**

**✅ Step 3: Copy Existing Data Into**Projects\_New

Run this **third** to migrate data over.

sql

Copy

SET IDENTITY\_INSERT dbo.Projects ON; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; SET IDENTITY\_INSERT dbo.Projects OFF;

👉 **Once confirmed, move to Step 4.**

**✅ Step 4: Check If Foreign Keys Are Blocking the Drop**

Run this **before Step 5** to find any foreign key constraints.

sql

Copy

SELECT name FROM sys.foreign\_keys WHERE referenced\_object\_id = OBJECT\_ID('dbo.Projects');

👉 **If this returns FK names, you must drop them using:**

sql

Copy

ALTER TABLE [dbo.OtherTable] DROP CONSTRAINT [FK\_YourForeignKeyName];

(Replace dbo.OtherTable and FK\_YourForeignKeyName with the actual values.)

**✅ Step 5: Drop the Old**Projects**Table**

Run this **fifth**, but **only if Step 4 confirms it’s safe to drop.**

sql

Copy

DROP TABLE dbo.Projects;

👉 **If SQL doesn’t allow it, let me know, and we’ll troubleshoot.**

**✅ Step 6: Rename**Projects\_New**to**Projects

Run this **sixth** to make it the official table.

sql

Copy

EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

👉 **Once confirmed, move to Step 7.**

**✅ Step 7: Insert P6 Data Into**dbo.Projects

Run this **seventh** to bring in new project data from P6.

sql

Copy

INSERT INTO dbo.Projects (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

👉 **If successful, final step!**

**✅ Step 8: Verify Everything**

Run this **last** to confirm the data is correct.

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If the projects appear correctly, 🎉 we’re done!**

**🔹 Summary**

🚀 **Run each step individually, checking for errors before moving to the next one.**  
💡 **If anything fails, stop and let me know—I’ll help troubleshoot!** 🔧🔥

Let me know how it goes, mate!

**You said:**

can we confirm the existence of dbo.Projects\_New, before we move on to step 3, please...and go step by step now....to get this right...time is running out.

**ChatGPT said:**

**Absolutely, mate!** ✅ Let’s **slow it down and go step by step** to make sure we get this **100% right** before moving forward.

**🔹 Step 1: Check If**dbo.Projects\_New**Already Exists**

Before we try inserting data, let’s confirm if the table was created successfully.

**✅ Run This Query**

sql

Copy

SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Projects\_New' AND TABLE\_SCHEMA = 'dbo';

👉 **If this returns a row,**Projects\_New**exists.**  
👉 **If it returns nothing, the table wasn’t created yet.**

**🔹 What You Need to Do Next**

✅ **Run the query above and tell me the result.**  
✅ **If**Projects\_New**exists, we move to Step 3 (inserting data).**  
✅ **If it doesn’t, we go back and fix it before continuing.**

Time is tight, so let’s **lock this down properly!** 🔧🔥 Let me know the result! 🚀

**You said:**

SQLP6DB dbo Projects\_New BASE TABLE,---right next step

**ChatGPT said:**

**Nice one, mate!** ✅ Projects\_New**exists**, so we can **move to Step 3** and insert data into it.

**🔹 Step 3: Insert Data from**dbo.Projects**into**dbo.Projects\_New

Now, we’ll **copy all existing project data** into Projects\_New.

**✅ Run This Query**

sql

Copy

SET IDENTITY\_INSERT dbo.Projects\_New ON; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects; SET IDENTITY\_INSERT dbo.Projects\_New OFF;

👉 **This ensures we retain all data before dropping**Projects**.**

**🔹 What You Need to Do Next**

✅ **Run the query and let me know if you get any errors.**  
✅ **Once it runs successfully, we’ll verify the data before moving to Step 4.**

🚀 **Let’s keep this rolling!** 🔧🔥

**You said:**

think we need to check the headers of dbo.Projects\_New before we insert any data...alot of underlined red errors are showing up in the that code.....whats the code to check headers exist first

**ChatGPT said:**

**Good thinking, mate!** ✅ Before inserting, let’s **verify the column headers** in dbo.Projects\_New to ensure they match what we need.

**🔹 Step 3.1: Check Column Headers in**dbo.Projects\_New

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Projects\_New' AND TABLE\_SCHEMA = 'dbo';

👉 **This will list all column names and their data types.**

**🔹 What You Need to Do Next**

✅ **Run the query and check if the columns match:**  
✔ ProjectID (INT)  
✔ ProjectName (NVARCHAR)  
✔ StartDate (DATE)  
✔ EndDate (DATE)

✅ **If the headers match, we move to inserting data.**  
✅ **If something is missing, we fix it before inserting.**

🚀 **Let me know what the query returns!** 🔧🔥

**You said:**

ProjectID int ProjectName nvarchar StartDate date EndDate date - thats the result from step 3.1 ProjectID cant be an INT, as its going to have alphacharacters in it, I would of thought like "DHBL" or "EC000515"

**ChatGPT said:**

**Great catch, mate!** ✅ Since ProjectID needs to **support alphanumeric values** like "DHBL" or "EC000515", we **must change it from**INT**to**NVARCHAR before inserting data.

**🔹 Step 3.2: Modify**ProjectID**to**NVARCHAR

Since ProjectID is currently INT, we **need to change its data type** before inserting data.

**✅ Run This Query to Alter the Column Type**

sql

Copy

ALTER TABLE dbo.Projects\_New ALTER COLUMN ProjectID NVARCHAR(50);

👉 **This will allow text-based Project IDs.**  
👉 We use NVARCHAR(50) to support both **letters and numbers**.

**🔹 What You Need to Do Next**

✅ **Run Step 3.2 to modify**ProjectID**.**  
✅ **Once done, re-run Step 3.1 to confirm**ProjectID**is now**NVARCHAR(50)**.**  
✅ **Let me know the result before we insert data!** 🚀🔥

**You said:**

Msg 5074, Level 16, State 1, Line 1 The object 'PK\_\_Projects\_\_761ABED077DE191C' is dependent on column 'ProjectID'. Msg 4922, Level 16, State 9, Line 1 ALTER TABLE ALTER COLUMN ProjectID failed because one or more objects access this column. Total execution time: 00:00:00.003

**ChatGPT said:**

**Got it, mate!** ✅ The error means that ProjectID is **already a primary key**, and SQL won’t let us change its data type while it's referenced by other constraints.

To fix this, we need to:  
1️⃣ **Drop the primary key constraint on**ProjectID  
2️⃣ **Change**ProjectID**to**NVARCHAR(50)  
3️⃣ **Recreate the primary key on**ProjectID

**🔹 Step 3.2 (Fixed) – Change**ProjectID**to**NVARCHAR

**✅ Step 1: Find the Primary Key Name**

Before we drop the primary key, we need to find its exact name.

sql

Copy

SELECT name FROM sys.key\_constraints WHERE type = 'PK' AND parent\_object\_id = OBJECT\_ID('dbo.Projects\_New');

👉 **This will return the primary key name (e.g.,**PK\_\_Projects\_\_761ABED077DE191C**).**

**✅ Step 2: Drop the Primary Key Constraint**

Once you get the primary key name from **Step 1**, replace [YourPrimaryKeyName] below and run:

sql

Copy

ALTER TABLE dbo.Projects\_New DROP CONSTRAINT [YourPrimaryKeyName];

👉 **This removes the primary key so we can modify the column.**

**✅ Step 3: Change**ProjectID**to**NVARCHAR(50)

sql

Copy

ALTER TABLE dbo.Projects\_New ALTER COLUMN ProjectID NVARCHAR(50) NOT NULL;

👉 **Now,**ProjectID**supports alphanumeric values.**

**✅ Step 4: Recreate the Primary Key**

sql

Copy

ALTER TABLE dbo.Projects\_New ADD CONSTRAINT PK\_Projects\_New PRIMARY KEY (ProjectID);

👉 **This ensures**ProjectID**remains unique after changing the data type.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to find the primary key name.**  
✅ **Run Step 2 to drop the primary key.**  
✅ **Run Step 3 to change**ProjectID**to**NVARCHAR(50)**.**  
✅ **Run Step 4 to recreate the primary key.**  
✅ **Re-run Step 3.1 to confirm**ProjectID**is now**NVARCHAR(50)**.**

🚀 **Let me know the results, and we’ll move forward!** 🔧🔥

**You said:**

PK\_Projects\_New all steps complete

**ChatGPT said:**

**Brilliant, mate!** ✅ Now that ProjectID is correctly set as NVARCHAR(50), we can **move to Step 3.3 and insert the data** into Projects\_New.

**🔹 Step 3.3: Insert Data from**dbo.Projects**into**dbo.Projects\_New

Since ProjectID is now alphanumeric, we can safely **copy all existing data over.**

**✅ Run This Query**

sql

Copy

INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT ProjectID, ProjectName, StartDate, EndDate FROM dbo.Projects;

👉 **If this runs successfully, we’ll verify the data before moving forward.**

**🔹 Step 3.4: Verify the Data in**Projects\_New

Once the data is inserted, let’s **check if everything is stored correctly.**

**✅ Run This Query**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects\_New;

👉 **If you see project data, 🎉 we’re all set to replace**Projects**!**

**🔹 What You Need to Do Next**

✅ **Run Step 3.3 to insert the data.**  
✅ **Run Step 3.4 to verify the data.**  
✅ **Let me know if any errors pop up before we move to the next step!** 🚀🔥

**You said:**

Started executing query at Line 1 (84 rows affected) Total execution time: 00:00:00.003 - seems to have worked, on Step 3.3, lets verify the data

**ChatGPT said:**

**Nice one, mate!** ✅ **84 rows inserted successfully** into Projects\_New. Now, let’s **verify the data** before we move forward.

**🔹 Step 3.4: Verify the Data in**Projects\_New

**✅ Run This Query**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects\_New;

👉 **This will show the first 10 rows to confirm everything looks correct.**

**🔹 What You Need to Do Next**

✅ **Run the query above and check if the data looks good.**  
✅ **If everything checks out, we move to Step 4 (replacing**Projects**).**

🚀 Let me know what you see, and we’ll keep rolling! 🔧🔥

**You said:**

3575 Unknown Project NULL NULL 3576 Unknown Project NULL NULL 3577 Unknown Project NULL NULL 3578 Unknown Project NULL NULL 3581 Unknown Project NULL NULL 3583 Unknown Project NULL NULL 3688 Unknown Project NULL NULL 3689 Unknown Project NULL NULL 3690 Unknown Project NULL NULL 3691 Unknown Project NULL NULL

**ChatGPT said:**

**Alright, mate!** ✅ The data **is there**, but it looks like we have **"Unknown Project" as the**ProjectName**and NULL**StartDate/EndDate**values.**

**🔹 What This Means**

1️⃣ ProjectID**was copied correctly** (e.g., 3575, 3576, etc.).  
2️⃣ **But**ProjectName**,**StartDate**, and**EndDate**didn’t come over correctly.**  
3️⃣ **We need to check where the data is coming from in**dbo.Projects**.**

**🔹 Step 3.5: Check the Data in**dbo.Projects**Before Moving Forward**

Let’s **verify what’s in**dbo.Projects to ensure we're copying the right values.

**✅ Run This Query**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If**Projects**also has "Unknown Project" and NULL dates, the issue is with the original data.**  
👉 **If**Projects**has correct names and dates, we need to re-run Step 3.3 more carefully.**

**🔹 What You Need to Do Next**

✅ **Run the query above and tell me what it returns.**  
✅ **If the data in**dbo.Projects**is correct, we’ll re-run the insert statement with a fix.**  
✅ **If**dbo.Projects**also has "Unknown Project", we need to fix the source data before moving forward.**

🚀 **Let me know what you find, and we’ll get this sorted!** 🔧🔥

**You said:**

3575 Unknown Project NULL NULL 3576 Unknown Project NULL NULL 3577 Unknown Project NULL NULL 3578 Unknown Project NULL NULL 3581 Unknown Project NULL NULL 3583 Unknown Project NULL NULL 3688 Unknown Project NULL NULL 3689 Unknown Project NULL NULL 3690 Unknown Project NULL NULL 3691 Unknown Project NULL NULL

**ChatGPT said:**

**Alright, mate!** ✅ Since dbo.Projects**also has "Unknown Project" and NULL dates**, the issue is **with the original data**—not the insert process.

Now, we need to **check where this data is coming from in P6** and correct it.

**🔹 Step 3.6: Check the Source Data in P6’s**PROJECT**Table**

Let’s verify if the **P6**PROJECT**table** has the correct names and dates.

**✅ Run This Query**

sql

Copy

SELECT proj\_id, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT WHERE proj\_id IN (3575, 3576, 3577, 3578, 3581, 3583, 3688, 3689, 3690, 3691);

👉 **This will show us what P6 actually has for these projects.**  
👉 If P6 also has **"Unknown Project" and NULL dates**, then the issue is with **how the data was stored in P6**.  
👉 If P6 **has the correct names and dates**, then we need to **adjust our insert query.**

**🔹 What You Need to Do Next**

✅ **Run the query above and tell me the results.**  
✅ **If P6 has correct data, we’ll fix the insert query.**  
✅ **If P6 also has bad data, we need to fix it there first before continuing.**

🚀 **Let’s get this right, mate! Let me know what P6 returns.** 🔧🔥

**You said:**

3575 CORP00591 2010-07-01 00:00:00.000 NULL 3576 CORP00424 2010-10-01 00:00:00.000 NULL 3577 CORP00307 2011-09-01 00:00:00.000 NULL 3578 CORP00103 2011-10-03 00:00:00.000 NULL 3581 CORP00768 2010-07-01 00:00:00.000 NULL 3583 CORP00118 2010-11-01 00:00:00.000 NULL 3688 CORP00712 2012-02-06 00:00:00.000 NULL 3689 CORP00595 2012-01-02 00:00:00.000 NULL 3690 CORP00852 2012-09-03 00:00:00.000 NULL 3691 CORP00384 2012-01-02 00:00:00.000 NULL - CORP00384, is the project id, that we need to reference against the other fields, ....3579 etc, looks like an SQL given project id, which is useless in reporting terms.

**ChatGPT said:**

**Got it, mate!** ✅ The real **Project ID should be**proj\_short\_name**(like CORP00384),** not the proj\_id (3575, etc.), which was just an internal SQL-generated value.

Now, we need to **correct our data in**Projects\_New so that:  
✔ ProjectID**stores**proj\_short\_name (e.g., CORP00384) instead of proj\_id (e.g., 3575).  
✔ **We retain**plan\_start\_date**as**StartDate**.**

**🔹 Step 3.7: Fix the Incorrect**ProjectID**Values**

Since dbo.Projects\_New currently has the wrong ProjectID, we need to:  
1️⃣ **Clear out the bad data**  
2️⃣ **Reinsert it with the correct values**

**✅ Step 1: Delete Incorrect Data From**Projects\_New

sql

Copy

DELETE FROM dbo.Projects\_New;

👉 **This removes the incorrect "Unknown Project" data.**

**✅ Step 2: Reinsert Data With Correct**ProjectID

sql

Copy

INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_short\_name, proj\_short\_name, plan\_start\_date, plan\_end\_date FROM PROJECT;

👉 **This makes**ProjectID**store**proj\_short\_name**(e.g.,**CORP00384**).**  
👉 **We also use**proj\_short\_name**as**ProjectName**for now, since P6 doesn't have long names.**

**✅ Step 3: Verify the Fix**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects\_New;

👉 **Now,**ProjectID**should look like**CORP00384**instead of**3575**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to delete the incorrect data.**  
✅ **Run Step 2 to reinsert the correct**ProjectID**values.**  
✅ **Run Step 3 to verify everything looks good.**

🚀 **Let me know how it goes, mate! This should fix our reporting issue.** 🔧🔥

**You said:**

AutoSys-Arcadia - B1 AutoSys-Arcadia - B1 2010-07-01 NULL AutoSys-Deerfield - B1 AutoSys-Deerfield - B1 2010-12-01 NULL AutoSys-Waterville - B1 AutoSys-Waterville - B1 2011-10-01 NULL Bay Main - B1 Bay Main - B1 2011-08-01 NULL CORP00103 CORP00103 2011-10-03 NULL CORP00118 CORP00118 2010-11-01 NULL CORP00307 CORP00307 2011-09-01 NULL CORP00384 CORP00384 2012-01-02 NULL CORP00424 CORP00424 2010-10-01 NULL CORP00591 CORP00591 2010-07-01 NULL

**ChatGPT said:**

**Perfect, mate!** ✅ Now, ProjectID is correctly set to proj\_short\_name, so we’re **ready to move forward**.

**🔹 What This Fix Achieved**

✔ **ProjectID now correctly shows**CORP00307**,**AutoSys-Arcadia - B1**, etc.**  
✔ **StartDate is correctly populated from P6 (**plan\_start\_date**).**  
✔ **EndDate is still NULL (since P6 has it as NULL).**

**🔹 Step 4: Replace**Projects**With**Projects\_New

Now that Projects\_New has the correct data, we **need to make it the main**Projects**table**.

**✅ Step 4.1: Drop the Old**Projects**Table**

Before renaming, we **need to drop the old**Projects**table**, but **only if no foreign key constraints block us**.

**Find Foreign Keys (If Any)**

sql

Copy

SELECT name FROM sys.foreign\_keys WHERE referenced\_object\_id = OBJECT\_ID('dbo.Projects');

👉 **If this returns FK names, we need to drop them first.**  
👉 **If nothing is returned, we can proceed directly to dropping the table.**

**Drop the Old**Projects**Table**

sql

Copy

DROP TABLE dbo.Projects;

👉 **This clears out the old incorrect table.**

**✅ Step 4.2: Rename**Projects\_New**to**Projects

sql

Copy

EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

👉 **Now, our fixed table is officially**Projects**.**

**✅ Step 4.3: Verify the Final Table**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If everything looks good, we’re done with this part!** 🎉

**🔹 What You Need to Do Next**

✅ **Run Step 4.1 to check for foreign keys and drop**Projects**.**  
✅ **Run Step 4.2 to rename**Projects\_New**to**Projects**.**  
✅ **Run Step 4.3 to verify that everything is correct.**

🚀 **Let me know how it goes, mate! We’re nearly there.** 🔧🔥

**You said:**

CORP00384, does have an finish date, not end date, ...of 30-01-2013. so think we missing something here.

**ChatGPT said:**

**Good spot, mate!** ✅ If CORP00384 has a **Finish Date (**30-01-2013**)** but it’s **not appearing in the**EndDate**column**, then we need to **find the correct column in P6 that stores Finish Dates**.

**🔹 Step 4.1: Check for the Correct "Finish Date" Column in P6**

The PROJECT table in P6 has **multiple date columns**, so let’s check which one actually holds the Finish Date.

**✅ Run This Query**

sql

Copy

SELECT proj\_id, proj\_short\_name, plan\_end\_date, scd\_end\_date, last\_recalc\_date FROM PROJECT WHERE proj\_short\_name = 'CORP00384';

👉 **This will check**plan\_end\_date**,**scd\_end\_date**, and**last\_recalc\_date**to see which one holds**30-01-2013**.**  
👉 **If**plan\_end\_date**is NULL, but**scd\_end\_date**has a value, we need to update our data insert process.**

**🔹 Step 4.2: Adjust the Insert Query to Use the Correct Finish Date**

If scd\_end\_date holds the correct Finish Date, we need to **update our insert query**.

**✅ Fix the Insert Query (Use**scd\_end\_date**if**plan\_end\_date**is NULL)**

sql

Copy

DELETE FROM dbo.Projects\_New; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_short\_name, proj\_short\_name, plan\_start\_date, COALESCE(plan\_end\_date, scd\_end\_date) *-- Use `scd\_end\_date` if `plan\_end\_date` is NULL* FROM PROJECT;

👉 **This ensures we get the correct Finish Date instead of NULL values.**

**🔹 Step 4.3: Verify the Data Again**

sql

Copy

SELECT \* FROM dbo.Projects\_New WHERE ProjectID = 'CORP00384';

👉 **Now,**EndDate**should correctly show**30-01-2013**.**

**🔹 What You Need to Do Next**

✅ **Run Step 4.1 to check which column holds the correct Finish Date.**  
✅ **If it’s**scd\_end\_date**, run Step 4.2 to fix the insert query.**  
✅ **Run Step 4.3 to verify the data is correct.**

🚀 **Let me know what Step 4.1 returns, and we’ll finalize this fix!** 🔧🔥

**You said:**

3691 CORP00384 NULL 2013-01-30 12:29:00.000 2009-12-31 00:00:00.000 - step 4.1 found a date.

**ChatGPT said:**

**Nice one, mate!** ✅ **Step 4.1 confirms that**scd\_end\_date**(2013-01-30) holds the correct Finish Date.**

Now, we need to **update our insert query** to use scd\_end\_date as the EndDate in Projects\_New.

**🔹 Step 4.2: Fix the Insert Query to Use**scd\_end\_date

Since scd\_end\_date holds the actual Finish Date, we need to **reinsert the data correctly**.

**✅ Run This Query to Fix the Data**

sql

Copy

DELETE FROM dbo.Projects\_New; INSERT INTO dbo.Projects\_New (ProjectID, ProjectName, StartDate, EndDate) SELECT proj\_short\_name, proj\_short\_name, plan\_start\_date, COALESCE(plan\_end\_date, scd\_end\_date) *-- Use `scd\_end\_date` if `plan\_end\_date` is NULL* FROM PROJECT;

👉 **This will fix**EndDate**for all projects, including**CORP00384**.**  
👉 **If**plan\_end\_date**is NULL, it will take**scd\_end\_date**instead.**

**🔹 Step 4.3: Verify the Data Again**

Once the insert is complete, let’s **check that**EndDate**is now correct**.

**✅ Run This Query**

sql

Copy

SELECT \* FROM dbo.Projects\_New WHERE ProjectID = 'CORP00384';

👉 **If**EndDate**now shows**2013-01-30**, we’re good to go!** 🎉

**🔹 What You Need to Do Next**

✅ **Run Step 4.2 to fix the insert query.**  
✅ **Run Step 4.3 to verify the data for**CORP00384**.**  
✅ **If**EndDate**is correct, we move forward with replacing**Projects**.**

🚀 **Let me know how it goes, mate! We’re nearly done!** 🔧🔥

**You said:**

CORP00384 CORP00384 2012-01-02 2013-01-30

**ChatGPT said:**

**Boom! 🎯 That’s exactly what we needed!** ✅ Now that Projects\_New has the **correct**ProjectID**,**StartDate**, and**EndDate, we can move on to **Step 5: Replacing**Projects**with**Projects\_New**.**

**🔹 Step 5: Replace**dbo.Projects**with**dbo.Projects\_New

Now, we make Projects\_New the **official**Projects**table**.

**✅ Step 5.1: Check for Foreign Keys Before Dropping**Projects

Before we drop Projects, we **must check for foreign key constraints** in other tables.

**Run This Query to Find Foreign Keys:**

sql

Copy

SELECT name FROM sys.foreign\_keys WHERE referenced\_object\_id = OBJECT\_ID('dbo.Projects');

👉 **If this returns FK names, we need to drop them first.**  
👉 **If nothing is returned, we can proceed directly to Step 5.2.**

**✅ Step 5.2: Drop the Old**Projects**Table**

If Step 5.1 **doesn’t return any foreign keys**, run this:

sql

Copy

DROP TABLE dbo.Projects;

👉 **If SQL blocks this due to foreign keys, let me know, and we’ll handle it.**

**✅ Step 5.3: Rename**Projects\_New**to**Projects

sql

Copy

EXEC sp\_rename 'dbo.Projects\_New', 'Projects';

👉 **Now,**Projects**contains the correct data.**

**✅ Step 5.4: Verify Everything**

sql

Copy

SELECT TOP 10 \* FROM dbo.Projects;

👉 **If the correct data appears, we’re DONE!** 🎉

**🔹 What You Need to Do Next**

✅ **Run Step 5.1 to check for foreign keys.**  
✅ **If there are no foreign keys, run Step 5.2 to drop**Projects**.**  
✅ **Run Step 5.3 to rename**Projects\_New**to**Projects**.**  
✅ **Run Step 5.4 to verify everything is correct.**

🚀 **Let me know how it goes, mate! We’re locking this down. 🔧🔥**

**You said:**

are you online and ready bud?

**ChatGPT said:**

Always here, mate! ✅ Locked in and ready to roll. 🔧🔥 Let’s get this done—what’s the next move? 🚀

**You said:**

well we were having some difficulties with the projects tables, but we need to get P6 data into the Resources table, can we do that

**ChatGPT said:**

**Got it, mate!** ✅ Now that Projects is sorted, let’s **move on to the**Resources**table** and pull in **P6 resource data**.

**🔹 Step 1: Check**dbo.Resources**Table Structure**

Before inserting data, let’s confirm **what columns exist** in dbo.Resources.

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **This will list all the column names and their data types.**  
👉 We need to **match these with P6’s**RSRC**table** before inserting data.

**🔹 Step 2: Check Where Resource Data is Stored in P6**

Now, let’s find out which columns P6 has in the RSRC table.

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'RSRC';

👉 **This will tell us which P6 columns we need to map to**dbo.Resources**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check the**Resources**table structure.**  
✅ **Run Step 2 to check the**RSRC**table in P6.**  
✅ **Send me both results so we can match columns and insert data properly.**

🚀 Let’s get this rolling, mate! 🔧🔥

**You said:**

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo'; this is step 1, ...however I also noticed that the resource\_ID is an INT, and it needs to be a VAR as well, as you might have a resource ID, like MS301, or LS31 etc....

**ChatGPT said:**

**Good catch, mate!** ✅ If ResourceID needs to store values like "MS301" or "LS31", we must **change it from**INT**to**NVARCHAR before inserting P6 data.

**🔹 Step 1.1: Modify**ResourceID**to**NVARCHAR

Since ResourceID is a **primary key**, we need to:  
1️⃣ **Find and drop the primary key**  
2️⃣ **Change**ResourceID**to**NVARCHAR(50)  
3️⃣ **Recreate the primary key**

**✅ Step 1.1.1: Find the Primary Key Name**

sql

Copy

SELECT name FROM sys.key\_constraints WHERE type = 'PK' AND parent\_object\_id = OBJECT\_ID('dbo.Resources');

👉 **This will return the primary key name (e.g.,**PK\_\_Resources\_\_XYZ**).**

**✅ Step 1.1.2: Drop the Primary Key Constraint**

Replace [YourPrimaryKeyName] with the actual name from Step 1.1.1 and run:

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT [YourPrimaryKeyName];

👉 **This removes the primary key so we can modify the column.**

**✅ Step 1.1.3: Change**ResourceID**to**NVARCHAR(50)

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN ResourceID NVARCHAR(50) NOT NULL;

👉 **Now,**ResourceID**supports both letters and numbers.**

**✅ Step 1.1.4: Recreate the Primary Key**

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **This ensures**ResourceID**stays unique.**

**🔹 Step 1.2: Verify the Column Change**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1.1.1 to find the primary key name.**  
✅ **Run Step 1.1.2 to drop the primary key.**  
✅ **Run Step 1.1.3 to modify**ResourceID**.**  
✅ **Run Step 1.1.4 to recreate the primary key.**  
✅ **Run Step 1.2 to verify that**ResourceID**is now**NVARCHAR(50)**.**

🚀 Once this is confirmed, we’ll pull in P6 resource data! 🔧🔥 Let me know how it goes!

**You said:**

PK\_\_Resource\_\_4ED1814F53B9D074 = Step 1.1.1

**ChatGPT said:**

**Nice one, mate!** ✅ Now that we have the primary key name **(**PK\_\_Resource\_\_4ED1814F53B9D074**)**, let’s **drop it and update the column type**.

**🔹 Step 1.1.2: Drop the Primary Key Constraint**

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_\_Resource\_\_4ED1814F53B9D074;

👉 **This removes the primary key so we can modify**ResourceID**.**

**🔹 Step 1.1.3: Change**ResourceID**to**NVARCHAR(50)

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN ResourceID NVARCHAR(50) NOT NULL;

👉 **This allows alphanumeric Resource IDs like**"MS301"**or**"LS31"**.**

**🔹 Step 1.1.4: Recreate the Primary Key on**ResourceID

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **Now,**ResourceID**is still a primary key but supports text values.**

**🔹 Step 1.2: Verify the Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1.1.2 to drop the primary key.**  
✅ **Run Step 1.1.3 to modify**ResourceID**.**  
✅ **Run Step 1.1.4 to recreate the primary key.**  
✅ **Run Step 1.2 to verify**ResourceID**is now**NVARCHAR(50)**.**

🚀 Once confirmed, we can move on to pulling in P6 resource data! 🔧🔥 Let me know how it goes!

**You said:**

Msg 3725, Level 16, State 0, Line 1 The constraint 'PK\_\_Resource\_\_4ED1814F53B9D074' is being referenced by table 'ScenarioData', foreign key constraint 'FK\_\_ScenarioD\_\_Resou\_\_0AFD888E'. Msg 3727, Level 16, State 0, Line 1 Could not drop constraint. See previous errors. Total execution time: 00:00:00.012 = Step 1.1.2

**ChatGPT said:**

**Got it, mate!** ✅ The error means ResourceID**is being referenced by**ScenarioData via a **foreign key constraint (**FK\_\_ScenarioD\_\_Resou\_\_0AFD888E**)**.

To fix this, we need to:  
1️⃣ **Drop the foreign key constraint (**FK\_\_ScenarioD\_\_Resou\_\_0AFD888E**)** in ScenarioData.  
2️⃣ **Drop the primary key (**PK\_\_Resource\_\_4ED1814F53B9D074**)** in Resources.  
3️⃣ **Modify**ResourceID**to**NVARCHAR(50)**.**  
4️⃣ **Recreate the primary key and foreign key.**

**🔹 Step 1.1.2 (Fixed) – Drop the Foreign Key in**ScenarioData

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ScenarioData DROP CONSTRAINT FK\_\_ScenarioD\_\_Resou\_\_0AFD888E;

👉 **This removes the link between**ScenarioData**and**Resources**, allowing us to proceed.**

**🔹 Step 1.1.3: Drop the Primary Key in**Resources

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_\_Resource\_\_4ED1814F53B9D074;

👉 **Now, we can safely modify**ResourceID**.**

**🔹 Step 1.1.4: Modify**ResourceID**to**NVARCHAR(50)

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN ResourceID NVARCHAR(50) NOT NULL;

👉 **Now,**ResourceID**supports alphanumeric values like**"MS301"**or**"LS31"**.**

**🔹 Step 1.1.5: Recreate the Primary Key on**ResourceID

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **This ensures**ResourceID**remains unique.**

**🔹 Step 1.1.6: Recreate the Foreign Key in**ScenarioData

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ScenarioData ADD CONSTRAINT FK\_ScenarioData\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **This restores the link between**ScenarioData**and**Resources**.**

**🔹 Step 1.2: Verify the Column Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1.1.2 to drop the foreign key.**  
✅ **Run Step 1.1.3 to drop the primary key.**  
✅ **Run Step 1.1.4 to modify**ResourceID**.**  
✅ **Run Step 1.1.5 to recreate the primary key.**  
✅ **Run Step 1.1.6 to recreate the foreign key.**  
✅ **Run Step 1.2 to verify that**ResourceID**is now**NVARCHAR(50)**.**

🚀 **Once this is done, we can bring in the P6 resource data!** 🔧🔥 Let me know how it goes, mate!

**You said:**

Msg 3725, Level 16, State 0, Line 1 The constraint 'PK\_\_Resource\_\_4ED1814F53B9D074' is being referenced by table 'ResourceAssignments', foreign key constraint 'FK\_\_ResourceA\_\_Resou\_\_072CF7AA'. Msg 3727, Level 16, State 0, Line 1 Could not drop constraint. See previous errors. on step 1.1.3

**ChatGPT said:**

**Alright, mate!** ✅ Now we know that ResourceID is also **referenced by**ResourceAssignments through the foreign key FK\_\_ResourceA\_\_Resou\_\_072CF7AA.

To fix this, we need to:  
1️⃣ **Drop both foreign key constraints (**ScenarioData**and**ResourceAssignments**).**  
2️⃣ **Drop the primary key in**Resources**.**  
3️⃣ **Modify**ResourceID**to**NVARCHAR(50)**.**  
4️⃣ **Recreate the primary key and foreign keys.**

**🔹 Step 1.1.2 (Updated) – Drop Foreign Keys in**ScenarioData**and**ResourceAssignments

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ScenarioData DROP CONSTRAINT FK\_\_ScenarioD\_\_Resou\_\_0AFD888E; ALTER TABLE dbo.ResourceAssignments DROP CONSTRAINT FK\_\_ResourceA\_\_Resou\_\_072CF7AA;

👉 **This removes the links blocking us from modifying**ResourceID**.**

**🔹 Step 1.1.3: Drop the Primary Key in**Resources

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_\_Resource\_\_4ED1814F53B9D074;

👉 **Now, we can safely modify**ResourceID**.**

**🔹 Step 1.1.4: Modify**ResourceID**to**NVARCHAR(50)

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN ResourceID NVARCHAR(50) NOT NULL;

👉 **Now,**ResourceID**supports alphanumeric values like**"MS301"**or**"LS31"**.**

**🔹 Step 1.1.5: Recreate the Primary Key on**ResourceID

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **This ensures**ResourceID**remains unique.**

**🔹 Step 1.1.6: Recreate Foreign Keys in**ScenarioData**and**ResourceAssignments

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ScenarioData ADD CONSTRAINT FK\_ScenarioData\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID); ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **This restores the links while keeping**ResourceID**as**NVARCHAR(50)**.**

**🔹 Step 1.2: Verify the Column Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1.1.2 to drop both foreign keys.**  
✅ **Run Step 1.1.3 to drop the primary key.**  
✅ **Run Step 1.1.4 to modify**ResourceID**.**  
✅ **Run Step 1.1.5 to recreate the primary key.**  
✅ **Run Step 1.1.6 to recreate the foreign keys.**  
✅ **Run Step 1.2 to verify that**ResourceID**is now**NVARCHAR(50)**.**

🚀 **Once this is done, we can load the P6 resource data!** 🔧🔥 Let me know how it goes, mate!

**You said:**

Started executing query at Line 1 Msg 3728, Level 16, State 1, Line 1 'FK\_\_ScenarioD\_\_Resou\_\_0AFD888E' is not a constraint. Msg 3727, Level 16, State 0, Line 1 Could not drop constraint. See previous errors. Total execution time: 00:00:00.001 step 1.1.2

**ChatGPT said:**

**Alright, mate!** ✅ Since SQL says **'FK\_\_ScenarioD\_\_Resou\_\_0AFD888E' is not a constraint**, we need to **double-check the actual foreign key name** before attempting to drop it.

**🔹 Step 1.1.2 (Fixed) – Find the Correct Foreign Key Names**

**✅ Run This Query to Get the Actual FK Names**

sql

Copy

SELECT fk.name AS ForeignKeyName, t.name AS TableName, c.name AS ColumnName FROM sys.foreign\_keys AS fk INNER JOIN sys.tables AS t ON fk.parent\_object\_id = t.object\_id INNER JOIN sys.columns AS c ON fk.parent\_object\_id = c.object\_id AND fk.parent\_column\_id = c.column\_id WHERE fk.referenced\_object\_id = OBJECT\_ID('dbo.Resources');

👉 **This will return all foreign keys referencing**dbo.Resources (including ScenarioData and ResourceAssignments).  
👉 **Look for the real foreign key names and send them to me.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the correct FK names.**  
✅ **Once we have them, I’ll give you the exact DROP commands.**

🚀 Let’s get this sorted properly, mate! 🔧🔥

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 7 Invalid column name 'parent\_column\_id'. Total execution time: 00:00:00.042

**ChatGPT said:**

**Alright, mate!** ✅ The error happened because sys.foreign\_keys **doesn’t store**parent\_column\_id**directly**. Let's **fix the query** and properly find the **foreign keys referencing**dbo.Resources.

**🔹 Step 1.1.2 (Fixed) – Find Correct Foreign Key Names**

**✅ Run This Query**

sql

Copy

SELECT fk.name AS ForeignKeyName, tp.name AS ParentTable, tr.name AS ReferencedTable FROM sys.foreign\_keys AS fk JOIN sys.tables AS tp ON fk.parent\_object\_id = tp.object\_id JOIN sys.tables AS tr ON fk.referenced\_object\_id = tr.object\_id WHERE tr.name = 'Resources';

👉 **This will list all foreign keys that reference**dbo.Resources**.**  
👉 **Look for the names related to**ScenarioData**and**ResourceAssignments**.**

**🔹 What You Need to Do Next**

✅ **Run the fixed query above.**  
✅ **Send me the correct foreign key names it returns.**  
✅ **Once we have them, I’ll give you the exact DROP commands.**

🚀 Let’s nail this down, mate! 🔧🔥

**You said:**

FK\_\_ResourceA\_\_Resou\_\_072CF7AA ResourceAssignments Resources

**ChatGPT said:**

**Nice one, mate!** ✅ Now that we have the correct foreign key name **(**FK\_\_ResourceA\_\_Resou\_\_072CF7AA**)**, we can proceed with **dropping it properly** before modifying ResourceID.

**🔹 Step 1.1.2 (Final) – Drop the Foreign Key in**ResourceAssignments

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ResourceAssignments DROP CONSTRAINT FK\_\_ResourceA\_\_Resou\_\_072CF7AA;

👉 **This removes the foreign key link between**ResourceAssignments**and**Resources**, so we can modify**ResourceID**.**

**🔹 Step 1.1.3: Drop the Primary Key in**Resources

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_\_Resource\_\_4ED1814F53B9D074;

👉 **Now, we can safely modify**ResourceID**.**

**🔹 Step 1.1.4: Modify**ResourceID**to**NVARCHAR(50)

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN ResourceID NVARCHAR(50) NOT NULL;

👉 **This allows**ResourceID**to store alphanumeric values like**"MS301"**or**"LS31"**.**

**🔹 Step 1.1.5: Recreate the Primary Key on**ResourceID

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **This ensures**ResourceID**remains unique.**

**🔹 Step 1.1.6: Recreate the Foreign Key in**ResourceAssignments

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **This restores the link between**ResourceAssignments**and**Resources**.**

**🔹 Step 1.2: Verify the Column Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1.1.2 to drop the foreign key.**  
✅ **Run Step 1.1.3 to drop the primary key.**  
✅ **Run Step 1.1.4 to modify**ResourceID**.**  
✅ **Run Step 1.1.5 to recreate the primary key.**  
✅ **Run Step 1.1.6 to recreate the foreign key.**  
✅ **Run Step 1.2 to verify that**ResourceID**is now**NVARCHAR(50)**.**

🚀 **Once this is done, we can load the P6 resource data!** 🔧🔥 Let me know how it goes!

**You said:**

Started executing query at Line 1 Msg 2749, Level 16, State 3, Line 1 Identity column 'ResourceID' must be of data type int, bigint, smallint, tinyint, or decimal or numeric with a scale of 0, unencrypted, and constrained to be nonnullable. Total execution time: 00:00:00.001 = Step 1.1.4

**ChatGPT said:**

**Got it, mate!** ✅ The error means that ResourceID**is currently set as an**IDENTITY**column**, and **SQL Server only allows**IDENTITY**on numeric types** (INT, BIGINT, etc.).

Since we **need**ResourceID**to be**NVARCHAR(50), we must **remove the**IDENTITY**property** before altering the column type.

**🔹 Step 1.1.4 (Fixed) – Remove**IDENTITY**and Change**ResourceID**to**NVARCHAR(50)

Since we **can’t directly remove**IDENTITY, we need to:  
1️⃣ **Create a new column (**ResourceID\_New**) as**NVARCHAR(50)**.**  
2️⃣ **Copy all**ResourceID**values into**ResourceID\_New**.**  
3️⃣ **Drop the old**ResourceID**column.**  
4️⃣ **Rename**ResourceID\_New**to**ResourceID**.**

**✅ Run This Query**

sql

Copy

*-- 1️⃣ Add a new column without IDENTITY* ALTER TABLE dbo.Resources ADD ResourceID\_New NVARCHAR(50) NOT NULL; *-- 2️⃣ Copy data from old ResourceID to new column* UPDATE dbo.Resources SET ResourceID\_New = CAST(ResourceID AS NVARCHAR(50)); *-- 3️⃣ Drop the foreign key temporarily (if it still exists)* ALTER TABLE dbo.ResourceAssignments DROP CONSTRAINT FK\_ResourceAssignments\_ResourceID; *-- 4️⃣ Drop the primary key* ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_Resources; *-- 5️⃣ Drop the old IDENTITY column* ALTER TABLE dbo.Resources DROP COLUMN ResourceID; *-- 6️⃣ Rename the new column to ResourceID* EXEC sp\_rename 'dbo.Resources.ResourceID\_New', 'ResourceID', 'COLUMN'; *-- 7️⃣ Recreate the primary key on the new ResourceID* ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID); *-- 8️⃣ Recreate the foreign key* ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **This removes**IDENTITY**, changes**ResourceID**to**NVARCHAR(50)**, and restores all constraints.**

**🔹 Step 1.2: Verify the Column Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1.1.4 (Fixed) to fully remove**IDENTITY**and change**ResourceID**to**NVARCHAR(50)**.**  
✅ **Run Step 1.2 to verify the column type.**

🚀 Once this is done, we can **load the P6 resource data!** 🔧🔥 Let me know how it goes!

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 5 Invalid column name 'ResourceID\_New'. Total execution time: 00:00:00.001

**ChatGPT said:**

**Alright, mate!** ✅ The error means ResourceID\_New**wasn’t created properly** in Step 1.1.4. **Let’s slow it down and do this properly, step by step.**

**🔹 Step 1.1.4 (Fixed) – Remove**IDENTITY**and Change**ResourceID**to**NVARCHAR(50)

Since **we can’t remove**IDENTITY**directly**, we need to:  
1️⃣ **Check if**ResourceID\_New**already exists.**  
2️⃣ **Create a new column (**ResourceID\_New**) as**NVARCHAR(50)**.**  
3️⃣ **Copy all**ResourceID**values into**ResourceID\_New**.**  
4️⃣ **Drop foreign keys (if needed).**  
5️⃣ **Drop the primary key.**  
6️⃣ **Drop the old**ResourceID**column.**  
7️⃣ **Rename**ResourceID\_New**to**ResourceID**.**  
8️⃣ **Recreate the primary key and foreign keys.**

**✅ Step 1: Check If**ResourceID\_New**Already Exists**

Before creating the new column, let’s **see if it already exists.**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo' AND COLUMN\_NAME = 'ResourceID\_New';

👉 **If this returns a row, the column already exists.**  
👉 **If it returns nothing, we can proceed to Step 2.**

**✅ Step 2: Add**ResourceID\_New**as**NVARCHAR(50)**(If It Doesn’t Exist)**

sql

Copy

ALTER TABLE dbo.Resources ADD ResourceID\_New NVARCHAR(50) NOT NULL;

👉 **This ensures we have a proper column before we copy data.**

**✅ Step 3: Copy Data from**ResourceID**to**ResourceID\_New

sql

Copy

UPDATE dbo.Resources SET ResourceID\_New = CAST(ResourceID AS NVARCHAR(50));

👉 **This moves all data to the new column while keeping it safe.**

**✅ Step 4: Drop Foreign Keys (If They Exist)**

sql

Copy

ALTER TABLE dbo.ResourceAssignments DROP CONSTRAINT FK\_ResourceAssignments\_ResourceID;

👉 **This prevents SQL from blocking the column drop.**  
👉 **If SQL throws an error saying the constraint doesn’t exist, ignore it and move on.**

**✅ Step 5: Drop the Primary Key from**ResourceID

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_Resources;

👉 **Now, we can remove**ResourceID**.**

**✅ Step 6: Drop the Old**ResourceID**Column**

sql

Copy

ALTER TABLE dbo.Resources DROP COLUMN ResourceID;

👉 **This clears out the old**IDENTITY**column.**

**✅ Step 7: Rename**ResourceID\_New**to**ResourceID

sql

Copy

EXEC sp\_rename 'dbo.Resources.ResourceID\_New', 'ResourceID', 'COLUMN';

👉 **Now, we have**ResourceID**as**NVARCHAR(50)**.**

**✅ Step 8: Recreate the Primary Key on**ResourceID

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **This ensures**ResourceID**remains unique.**

**✅ Step 9: Recreate the Foreign Key in**ResourceAssignments

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **This restores the relationship between**ResourceAssignments**and**Resources**.**

**🔹 Step 1.2: Verify the Column Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 1 to check if**ResourceID\_New**already exists.**  
✅ **If it doesn’t, run Steps 2-9 in order.**  
✅ **Run Step 1.2 to verify**ResourceID**is now**NVARCHAR(50)**.**

🚀 **Once this is done, we can finally load the P6 resource data!** 🔧🔥 Let me know how it goes!

**You said:**

step 2, = Started executing query at Line 1 Msg 4901, Level 16, State 1, Line 1 ALTER TABLE only allows columns to be added that can contain nulls, or have a DEFAULT definition specified, or the column being added is an identity or timestamp column, or alternatively if none of the previous conditions are satisfied the table must be empty to allow addition of this column. Column 'ResourceID\_New' cannot be added to non-empty table 'Resources' because it does not satisfy these conditions. Total execution time: 00:00:00.004

**ChatGPT said:**

**Got it, mate!** ✅ The error means that **SQL Server won’t let us add a**NOT NULL**column (**ResourceID\_New**) to a table that already has data**, unless we provide a **default value** or allow NULL temporarily.

To **fix this**, we’ll:  
1️⃣ **Add**ResourceID\_New**as**NVARCHAR(50) NULL**first**  
2️⃣ **Copy data from**ResourceID**into**ResourceID\_New  
3️⃣ **Change**ResourceID\_New**to**NOT NULL**after data is in**

**🔹 Step 2 (Fixed) – Add**ResourceID\_New**With**NULL**Allowed**

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD ResourceID\_New NVARCHAR(50) NULL;

👉 **We allow**NULL**for now so SQL doesn’t complain.**

**🔹 Step 3: Copy Data from**ResourceID**to**ResourceID\_New

**✅ Run This Query**

sql

Copy

UPDATE dbo.Resources SET ResourceID\_New = CAST(ResourceID AS NVARCHAR(50));

👉 **This transfers all existing**ResourceID**values into**ResourceID\_New**.**

**🔹 Step 4: Change**ResourceID\_New**to**NOT NULL

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN ResourceID\_New NVARCHAR(50) NOT NULL;

👉 **Now that all rows have data, we make**ResourceID\_New**NOT NULL.**

**🔹 Step 5 (Continue with the Plan)**

Now you can **continue with Steps 4-9 from my last message**:  
✅ Drop foreign keys  
✅ Drop the primary key  
✅ Drop the old ResourceID  
✅ Rename ResourceID\_New to ResourceID  
✅ Recreate the primary key  
✅ Recreate the foreign key

**🔹 What You Need to Do Next**

✅ **Run Step 2 (Fixed) to add**ResourceID\_New**as**NULL**.**  
✅ **Run Step 3 to copy data into**ResourceID\_New**.**  
✅ **Run Step 4 to make**ResourceID\_NewNOT NULL**.**  
✅ **Then continue with the other steps as planned.**

🚀 **Let me know how it goes, mate! We’re getting close! 🔧🔥**

**You said:**

ok, all good on those, 242 rows affected, so on with the next, please pause after each step for feedback ok bud

**ChatGPT said:**

**Brilliant, mate!** ✅ **242 rows copied successfully**—we’re on the right track. 🔥

Now, let's **continue carefully** and take it **one step at a time**.

**🔹 Step 5: Drop the Foreign Key in**ResourceAssignments

Now that ResourceID\_New is ready, we need to **remove the foreign key constraint** in ResourceAssignments so SQL doesn’t block us from dropping the old ResourceID.

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ResourceAssignments DROP CONSTRAINT FK\_\_ResourceA\_\_Resou\_\_072CF7AA;

👉 **This removes the foreign key linking**ResourceAssignments**to**Resources**.**

**🔹 What You Need to Do Next**

✅ **Run Step 5 and confirm if it executed successfully.**  
✅ **If you get an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll move to dropping the primary key next.**

🚀 **Let me know how it goes, mate!** 🔧🔥

**You said:**

Msg 3728, Level 16, State 1, Line 1 'FK\_\_ResourceA\_\_Resou\_\_072CF7AA' is not a constraint. Msg 3727, Level 16, State 0, Line 1 Could not drop constraint. See previous errors.

**ChatGPT said:**

**Alright, mate!** ✅ SQL says **'FK\_\_ResourceA\_\_Resou\_\_072CF7AA' is not a constraint**, which means the actual foreign key name is different.

Let’s **double-check the correct foreign key name** before proceeding.

**🔹 Step 5.1: Find the Actual Foreign Key Name**

**✅ Run This Query**

sql

Copy

SELECT fk.name AS ForeignKeyName, tp.name AS ParentTable, tr.name AS ReferencedTable FROM sys.foreign\_keys AS fk JOIN sys.tables AS tp ON fk.parent\_object\_id = tp.object\_id JOIN sys.tables AS tr ON fk.referenced\_object\_id = tr.object\_id WHERE tr.name = 'Resources';

👉 **This will list all foreign keys that reference**dbo.Resources**.**  
👉 **Look for the one related to**ResourceAssignments**.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the correct FK name.**  
✅ **Once we have the correct name, I’ll give you the exact DROP command.**

🚀 Let’s get this sorted properly, mate! 🔧🔥

**You said:**

that came back blank, just with headers ForeignKeyname, ParentTable, ReferencedTable

**ChatGPT said:**

**Alright, mate!** ✅ Since the query **returned nothing**, that means **there is no active foreign key linking**ResourceAssignments**to**Resources**anymore.**

This is **good news** because it means we can **skip Step 5** and move straight to **dropping the primary key** from Resources.

**🔹 Step 6: Drop the Primary Key from**Resources

Now that the foreign key **isn’t blocking us**, let’s drop the **primary key on**ResourceID so we can replace it.

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP CONSTRAINT PK\_Resources;

👉 **This will remove the primary key from**ResourceID**, allowing us to rename**ResourceID\_New**.**

**🔹 What You Need to Do Next**

✅ **Run Step 6 and confirm if it executed successfully.**  
✅ **If SQL returns an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll move on to dropping the old**ResourceID**.**

🚀 **Let me know how it goes, mate!** 🔧🔥

**You said:**

Started executing query at Line 1 Msg 3728, Level 16, State 1, Line 1 'PK\_Resources' is not a constraint. Msg 3727, Level 16, State 0, Line 1 Could not drop constraint. See previous errors. Total execution time: 00:00:00.001

**ChatGPT said:**

**Alright, mate!** ✅ Since SQL says **'PK\_Resources' is not a constraint**, that means **the actual primary key name is different.**

Let’s **find the correct primary key name** before proceeding.

**🔹 Step 6.1: Find the Actual Primary Key Name**

**✅ Run This Query**

sql

Copy

SELECT name FROM sys.key\_constraints WHERE type = 'PK' AND parent\_object\_id = OBJECT\_ID('dbo.Resources');

👉 **This will return the actual primary key name for**Resources**.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the correct PK name.**  
✅ **Once we have the correct name, I’ll give you the exact DROP command.**

🚀 **Let’s get this sorted properly, mate! 🔧🔥**

**You said:**

Just comes up blank, with a Header, NAME

**ChatGPT said:**

**Alright, mate!** ✅ Since the query returned **nothing**, that means Resources**doesn’t have a primary key at all right now.**

This is **good news** because it means we can **skip Step 6** and move straight to **dropping the old**ResourceID**column.**

**🔹 Step 7: Drop the Old**ResourceID**Column**

Now that there’s **nothing blocking us**, let’s **remove the old**ResourceID (which still has the IDENTITY property).

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP COLUMN ResourceID;

👉 **This removes the old**ResourceID**column, leaving only**ResourceID\_New**.**

**🔹 What You Need to Do Next**

✅ **Run Step 7 and confirm if it executed successfully.**  
✅ **If SQL returns an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll move on to renaming**ResourceID\_New**to**ResourceID**.**

🚀 **Let me know how it goes, mate!** 🔧🔥

**You said:**

that worked....completed no errors

**ChatGPT said:**

**Brilliant, mate!** ✅ Now that the old ResourceID is gone, we can **rename**ResourceID\_New**to**ResourceID so everything is back to normal.

**🔹 Step 8: Rename**ResourceID\_New**to**ResourceID

**✅ Run This Query**

sql

Copy

EXEC sp\_rename 'dbo.Resources.ResourceID\_New', 'ResourceID', 'COLUMN';

👉 **This renames**ResourceID\_New**back to**ResourceID**, but now as**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 8 and confirm if it executed successfully.**  
✅ **If SQL returns an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll move on to recreating the primary key.**

🚀 **Let me know how it goes, mate!** 🔧🔥

**You said:**

step 8 = Started executing query at Line 1 Caution: Changing any part of an object name could break scripts and stored procedures. Total execution time: 00:00:00.182

**ChatGPT said:**

**Nice one, mate!** ✅ The **caution message is normal**—it’s just a warning in case other scripts reference ResourceID. **No errors means it worked!** 🎯

Now, let’s **recreate the primary key on**ResourceID to ensure it stays unique.

**🔹 Step 9: Recreate the Primary Key on**ResourceID

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD CONSTRAINT PK\_Resources PRIMARY KEY (ResourceID);

👉 **This ensures that**ResourceID**remains unique and indexed.**

**🔹 What You Need to Do Next**

✅ **Run Step 9 and confirm if it executed successfully.**  
✅ **If SQL returns an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll move on to restoring the foreign key.**

🚀 **Let me know how it goes, mate!** 🔧🔥

**You said:**

that returned all complete, no errors

**ChatGPT said:**

**Perfect, mate!** ✅ ResourceID**is now a primary key again**, and everything is back in place. 🎯

Now, let’s **restore the foreign key in**ResourceAssignments so the table relationships are fully functional.

**🔹 Step 10: Recreate the Foreign Key in**ResourceAssignments

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **This restores the link between**ResourceAssignments**and**Resources**.**

**🔹 What You Need to Do Next**

✅ **Run Step 10 and confirm if it executed successfully.**  
✅ **If SQL returns an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll move on to verifying the final setup.**

🚀 **Let me know how it goes, mate!** 🔧🔥

**You said:**

Started executing query at Line 1 Msg 1778, Level 16, State 0, Line 1 Column 'dbo.Resources.ResourceID' is not the same data type as referencing column 'ResourceAssignments.ResourceID' in foreign key 'FK\_ResourceAssignments\_ResourceID'. Msg 1750, Level 16, State 1, Line 1 Could not create constraint or index. See previous errors. Total execution time: 00:00:00.006

**ChatGPT said:**

**Alright, mate!** ✅ The error means that ResourceID**in**ResourceAssignments**is still the old data type (**INT**)**, while ResourceID in Resources is now NVARCHAR(50).

To **fix this**, we need to:  
1️⃣ **Check**ResourceID**data type in**ResourceAssignments  
2️⃣ **Modify**ResourceID**in**ResourceAssignments**to**NVARCHAR(50)  
3️⃣ **Recreate the foreign key constraint**

**🔹 Step 10.1: Check**ResourceID**Data Type in**ResourceAssignments

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ResourceAssignments' AND COLUMN\_NAME = 'ResourceID';

👉 **If it returns**INT**, we need to change it to**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Step 10.1 and send me the result.**  
✅ **If**ResourceID**is**INT**, we’ll update it to**NVARCHAR(50)**.**

🚀 **Let me know what it says, mate!** 🔧🔥

**You said:**

ResourceID int

**ChatGPT said:**

**Got it, mate!** ✅ Since ResourceID in ResourceAssignments is still INT, we need to **convert it to**NVARCHAR(50) so it matches dbo.Resources.ResourceID.

**🔹 Step 10.2: Convert**ResourceID**in**ResourceAssignments**to**NVARCHAR(50)

Since ResourceID is **part of a foreign key**, we need to:  
1️⃣ **Drop the foreign key (if it exists, just in case).**  
2️⃣ **Create a new column**ResourceID\_New**as**NVARCHAR(50)**.**  
3️⃣ **Copy data from the old**ResourceID**to**ResourceID\_New**.**  
4️⃣ **Drop the old**ResourceID**column.**  
5️⃣ **Rename**ResourceID\_New**to**ResourceID**.**  
6️⃣ **Recreate the foreign key constraint.**

**✅ Step 1: Drop the Foreign Key (If It Exists)**

sql

Copy

ALTER TABLE dbo.ResourceAssignments DROP CONSTRAINT FK\_\_ResourceA\_\_Resou\_\_072CF7AA;

👉 **If this errors out saying it doesn’t exist, ignore the error and move to the next step.**

**✅ Step 2: Add a New**ResourceID\_New**Column**

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD ResourceID\_New NVARCHAR(50) NULL;

👉 **We allow**NULL**for now to avoid errors.**

**✅ Step 3: Copy Data from Old**ResourceID**to**ResourceID\_New

sql

Copy

UPDATE dbo.ResourceAssignments SET ResourceID\_New = CAST(ResourceID AS NVARCHAR(50));

👉 **This transfers all existing**ResourceID**values.**

**✅ Step 4: Drop the Old**ResourceID**Column**

sql

Copy

ALTER TABLE dbo.ResourceAssignments DROP COLUMN ResourceID;

👉 **Now the old**INT**column is gone.**

**✅ Step 5: Rename**ResourceID\_New**to**ResourceID

sql

Copy

EXEC sp\_rename 'dbo.ResourceAssignments.ResourceID\_New', 'ResourceID', 'COLUMN';

👉 **Now we have**ResourceID**as**NVARCHAR(50)**.**

**✅ Step 6: Recreate the Foreign Key**

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **Now, the relationship is fully restored!**

**🔹 Step 10.3: Verify the Column Change**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'ResourceAssignments' AND COLUMN\_NAME = 'ResourceID';

👉 **Confirm that**ResourceID**is now**NVARCHAR(50)**.**

**🔹 What You Need to Do Next**

✅ **Run Steps 1-6 in order.**  
✅ **Run Step 10.3 to verify**ResourceID**is now**NVARCHAR(50)**.**  
✅ **Let me know if there are any errors or if all goes well!**

🚀 **We’re so close, mate! Let’s get this sorted. 🔧🔥**

**You said:**

Step 1 = Started executing query at Line 1 Msg 3728, Level 16, State 1, Line 1 'FK\_\_ResourceA\_\_Resou\_\_072CF7AA' is not a constraint. Msg 3727, Level 16, State 0, Line 1 Could not drop constraint. See previous errors. Total execution time: 00:00:00.001

**ChatGPT said:**

**Alright, mate!** ✅ Since SQL says **'FK\_\_ResourceA\_\_Resou\_\_072CF7AA' is not a constraint**, that means **there is no foreign key currently linking**ResourceAssignments**to**Resources.

This is **good news** because it means we can **skip Step 1** and move directly to **Step 2: Adding**ResourceID\_New.

**🔹 Step 10.2 (Continue) – Add**ResourceID\_New**and Convert Data**

**✅ Step 2: Add**ResourceID\_New**as**NVARCHAR(50)

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD ResourceID\_New NVARCHAR(50) NULL;

👉 **This allows us to copy data before making it**NOT NULL**.**

**✅ Step 3: Copy Data from Old**ResourceID**to**ResourceID\_New

sql

Copy

UPDATE dbo.ResourceAssignments SET ResourceID\_New = CAST(ResourceID AS NVARCHAR(50));

👉 **This moves all values from**ResourceID**into the new column.**

**✅ Step 4: Drop the Old**ResourceID**Column**

sql

Copy

ALTER TABLE dbo.ResourceAssignments DROP COLUMN ResourceID;

👉 **Now the old**INT**column is gone.**

**✅ Step 5: Rename**ResourceID\_New**to**ResourceID

sql

Copy

EXEC sp\_rename 'dbo.ResourceAssignments.ResourceID\_New', 'ResourceID', 'COLUMN';

👉 **Now we have**ResourceID**as**NVARCHAR(50)**.**

**✅ Step 6: Recreate the Foreign Key**

sql

Copy

ALTER TABLE dbo.ResourceAssignments ADD CONSTRAINT FK\_ResourceAssignments\_ResourceID FOREIGN KEY (ResourceID) REFERENCES dbo.Resources(ResourceID);

👉 **Now, the relationship is fully restored!**

**🔹 What You Need to Do Next**

✅ **Run Steps 2-6 in order.**  
✅ **Let me know if there are any errors or if all goes well!**

🚀 **We’re nearly done, mate! 🔧🔥**

**You said:**

all completed all good....

**ChatGPT said:**

**Absolutely smashing it, mate!** ✅ **All steps completed with no errors**—that’s what I like to hear! 🎯🔥

Now, let’s do a final check to make sure everything is set up properly.

**🔹 Step 11: Verify the Final Setup**

**✅ Run This Query to Check**ResourceID**in Both Tables**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME IN ('Resources', 'ResourceAssignments') AND COLUMN\_NAME = 'ResourceID';

👉 **This should return**NVARCHAR(50)**for both tables.**  
👉 **If both match, everything is correctly linked.**

**✅ Check the Foreign Key Exists**

sql

Copy

SELECT fk.name AS ForeignKeyName, t.name AS TableName FROM sys.foreign\_keys AS fk JOIN sys.tables AS t ON fk.parent\_object\_id = t.object\_id WHERE fk.referenced\_object\_id = OBJECT\_ID('dbo.Resources');

👉 **This should show that**ResourceAssignments**is correctly linked to**Resources**.**

**🔹 What You Need to Do Next**

✅ **Run the queries above and confirm that everything is set correctly.**  
✅ **If anything doesn’t match, let me know, and we’ll sort it out.**  
✅ **If all looks good, we are DONE with this part!** 🎉

🚀 **Let me know how the checks go, mate! 🔧🔥**

**You said:**

that all looks good

**ChatGPT said:**

**Brilliant, mate!** ✅ **All checks passed**—that means **we’re done with the Resources table!** 🎉🔥

Now, we can **finally bring in the P6 resource data** and move forward with the next steps.

**🔹 What’s Next?**

Now that ResourceID is correctly formatted, do you want to:  
1️⃣ **Import resource data from P6 into**Resources**?**  
2️⃣ **Start working on another table (like**ScenarioData**)?**  
3️⃣ **Build reports/dashboards from this data?**

Let me know what’s next, mate! 🚀🔧🔥

**You said:**

lets import P6 data into Resources table, and then we can output to a report.

**ChatGPT said:**

**Sounds like a solid plan, mate!** ✅ **First, we’ll import P6 resource data into**Resources**, then move on to reporting.**

**🔹 Step 1: Identify P6 Data for Import**

Before importing, let’s **see what resource data is available in P6’s**RSRC**table**.

**✅ Run This Query to View P6 Resource Data**

sql

Copy

SELECT TOP 10 \* FROM RSRC;

👉 **This will show a sample of what’s inside**RSRC**so we can match columns.**  
👉 **Once we know what’s available, we’ll write an**INSERT**statement to load it into**Resources**.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the results.**  
✅ **Let me know which P6 columns we need to map to**dbo.Resources**.**

🚀 **Once we have this info, we’ll write the import query! 🔧🔥**

**You said:**

1158 6564 1 N Y RT\_Labor N Y Y MathisL Lane Mathis, CIO 4443 60 30 NULL 91 NULL 1 NULL NULL 1.500 0.25000000 Mf0WzZUG1BGyHgBQ2rH1XR QT\_Year NULL 555-555-0124 NULL NULL NULL NULL NULL <HTML><BODY bgColo NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1159 178 50006 N Y RT\_Labor N Y Y AndersonG Glen Anderson, VP Development 1158 30 30 NULL NULL NULL 1 NULL NULL 1.500 0.25000000 Nv0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <H NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1171 178 20812 Y Y RT\_Labor N Y Y VincentI Ian Vincent 1159 30 30 NULL NULL 755 1 NULL NULL 1.500 1.00000000 Sv0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL NULL NULL 2010-10-13 12:59:22.000 NULL admin NULL NULL NULL NULL 1172 178 50106 N Y RT\_Labor N Y Y CharlesM Mandy Charles, VP IT Ops 1158 30 30 NULL NULL 756 1 NULL NULL 1.500 1.00000000 Tf0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <H NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1177 178 20815 Y Y RT\_Labor N Y Y ZhuS Shannon Zhu 1159 30 30 NULL NULL 753 1 NULL NULL 1.500 1.00000000 Of0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <HTML><BODY bgColor=#ffffff>< NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1183 178 401 Y Y RT\_Labor N Y Y LaffertyV Vanessa Lafferty 1172 30 30 NULL NULL 756 1 NULL NULL 1.500 1.00000000 Q/0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL NULL NULL 2010-02-08 17:59:02.000 NULL admin NULL NULL NULL NULL 1184 178 20814 Y Y RT\_Labor N Y Y BennettC Carina Bennett 1159 30 30 NULL NULL 755 1 NULL NULL 1.500 1.00000000 RP0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <H NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1185 178 501 Y Y RT\_Labor N Y Y PaxsonD Dan Paxson 1172 30 30 NULL NULL 756 1 NULL NULL 1.500 1.00000000 Qf0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <H NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1187 178 21011 Y Y RT\_Labor N Y Y SinghD Deepak Singh 1159 30 30 NULL NULL 753 1 NULL NULL 1.500 1.00000000 Rf0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <H NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL 1188 178 20811 N Y RT\_Labor N Y Y SharpeD Dan Sharpe 1159 30 30 NULL NULL 755 1 NULL NULL 1.500 0.75000000 SP0WzZUG1BGyHgBQ2rH1XR QT\_Hour NULL NULL NULL NULL NULL NULL NULL <HTML><BODY bgColor=#ffffff><P> NULL 2014-08-27 19:06:43.000 NULL NotPrmUser NULL NULL NULL NULL = resource\_short\_name, is the column we need for Resource\_ID in P6, just to be made aware ...but that all looks good.

**ChatGPT said:**

**Perfect, mate!** ✅ Now that we’ve identified resource\_short\_name as the correct field for ResourceID, let’s get this P6 data into dbo.Resources.

**🔹 Step 2: Insert P6 Resource Data into**dbo.Resources

Now, we’ll **map P6’s**RSRC**columns to**dbo.Resources. Here’s the best match based on the data you sent:

| **P6 RSRC Column** | **dbo.Resources Column** | **Description** |
| --- | --- | --- |
| resource\_short\_name | ResourceID | Unique resource identifier (primary key) |
| resource\_name | ResourceName | Full resource name |
| email\_addr | Email | Resource's email (if applicable) |
| phone\_num | Phone | Contact number (if applicable) |
| role\_id | RoleID | Assigned role |
| parent\_rsrc\_id | ParentResourceID | If resource is part of a hierarchy |
| rsrc\_type | ResourceType | Type of resource (Labor, Material, etc.) |
| act\_rate\_type | RateType | Hourly, daily, or other billing rate |

**✅ Run This Query to Insert P6 Data**

sql

Copy

INSERT INTO dbo.Resources (ResourceID, ResourceName, Email, Phone, RoleID, ParentResourceID, ResourceType, RateType) SELECT resource\_short\_name AS ResourceID, resource\_name AS ResourceName, email\_addr AS Email, phone\_num AS Phone, role\_id AS RoleID, parent\_rsrc\_id AS ParentResourceID, rsrc\_type AS ResourceType, act\_rate\_type AS RateType FROM RSRC;

👉 **This will insert all available resources from P6 into**dbo.Resources**.**

**🔹 What You Need to Do Next**

✅ **Run the query above and confirm how many rows were inserted.**  
✅ **If you get an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll verify the data before moving to reporting.**

🚀 **Let me know how it goes, mate! 🔧🔥**

**You said:**

Msg 207, Level 16, State 1, Line 3 Invalid column name 'resource\_short\_name'. Msg 207, Level 16, State 1, Line 4 Invalid column name 'resource\_name'. Msg 207, Level 16, State 1, Line 6 Invalid column name 'phone\_num'. Msg 207, Level 16, State 1, Line 10 Invalid column name 'act\_rate\_type'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'Email'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'Phone'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'RoleID'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'ParentResourceID'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'ResourceType'. Msg 207, Level 16, State 1, Line 1 Invalid column name 'RateType'. Total execution time: 00:00:00.002

**ChatGPT said:**

**Alright, mate!** ✅ The error means that some column names **don’t match exactly** in either the **P6**RSRC**table** or **our**dbo.Resources**table**.

Let’s **double-check the correct column names in both tables** before proceeding.

**🔹 Step 2.1: Get the Exact Column Names from P6 (**RSRC**)**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'RSRC' AND TABLE\_SCHEMA = 'dbo';

👉 **This will return the exact column names in P6’s**RSRC**table.**  
👉 **Look for anything similar to**resource\_short\_name**,**resource\_name**, etc.**

**🔹 Step 2.2: Get the Exact Column Names from**dbo.Resources

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **This will return the exact column names in**dbo.Resources**.**  
👉 **Compare these with what we are trying to insert.**

**🔹 What You Need to Do Next**

✅ **Run Step 2.1 and send me the correct P6 column names.**  
✅ **Run Step 2.2 and send me the correct**dbo.Resources**column names.**  
✅ **Once we match them properly, I’ll fix the**INSERT**statement.**

🚀 **Let’s get this sorted properly, mate! 🔧🔥**

**You said:**

step 2.1 = rsrc\_id clndr\_id rsrc\_seq\_num timesheet\_flag active\_flag rsrc\_type auto\_compute\_act\_flag ot\_flag def\_cost\_qty\_link\_flag rsrc\_short\_name rsrc\_name parent\_rsrc\_id xfer\_complete\_day\_cnt xfer\_notstart\_day\_cnt ts\_approve\_user\_id user\_id role\_id curr\_id last\_checksum shift\_id ot\_factor def\_qty\_per\_hr guid cost\_qty\_type office\_phone other\_phone employee\_code rsrc\_title\_name email\_addr unit\_id pobs\_id rsrc\_notes intg\_type update\_date location\_id update\_user create\_date create\_user delete\_session\_id delete\_date

**ChatGPT said:**

**Nice one, mate!** ✅ Now we have the exact **P6**RSRC**column names**—this will help us fix the INSERT query.

Next, let’s check the **column names from**dbo.Resources so we can match them properly.

**🔹 Step 2.2: Get Exact Column Names from**dbo.Resources

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND TABLE\_SCHEMA = 'dbo';

👉 **Send me the result so we can compare and fix the insert query.**

🚀 **Let’s get this right, mate! 🔧🔥**

**You said:**

Step 2.2 = ResourceName Role MaxUnits CalendarID ResourceID

**ChatGPT said:**

**Perfect, mate!** ✅ Now that we have both RSRC and dbo.Resources columns, let’s correctly map them for the INSERT query.

**🔹 Corrected Column Mapping**

Here’s how the **P6**RSRC**table** matches up with dbo.Resources:

| **P6**RSRC**Column** | **dbo.Resources Column** | **Description** |
| --- | --- | --- |
| rsrc\_short\_name | ResourceID | Unique resource identifier (primary key) |
| rsrc\_name | ResourceName | Full resource name |
| role\_id | Role | Assigned role |
| def\_qty\_per\_hr | MaxUnits | Default units per hour |
| clndr\_id | CalendarID | Assigned calendar |

**🔹 Step 3: Insert P6 Resource Data into**dbo.Resources

**✅ Run This Query**

sql

Copy

INSERT INTO dbo.Resources (ResourceID, ResourceName, Role, MaxUnits, CalendarID) SELECT rsrc\_short\_name AS ResourceID, rsrc\_name AS ResourceName, role\_id AS Role, def\_qty\_per\_hr AS MaxUnits, clndr\_id AS CalendarID FROM RSRC;

👉 **This will correctly insert the P6 resource data into**dbo.Resources**.**

**🔹 What You Need to Do Next**

✅ **Run the query above and confirm how many rows were inserted.**  
✅ **If you get an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll verify the data before moving to reporting.**

🚀 **Let me know how it goes, mate! 🔧🔥**

**You said:**

264 rows affected, that worked...are we done?

**ChatGPT said:**

**Absolutely nailed it, mate!** ✅ **264 rows successfully inserted**—that means **P6 resource data is now fully loaded into**dbo.Resources**!** 🎉🔥

**🔹 Final Verification**

Before we move on to reporting, let’s **quickly check that the data looks good.**

**✅ Run This Query**

sql

Copy

SELECT TOP 10 \* FROM dbo.Resources;

👉 **This will show a sample of what’s inside**dbo.Resources**now.**

**🔹 What’s Next?**

Now that the **data is in**, do you want to:  
1️⃣ **Generate reports in SQL (aggregations, availability, etc.)?**  
2️⃣ **Export the data to Excel for analysis?**  
3️⃣ **Load it into Power BI for dynamic dashboards?**

Let me know what’s next, mate! 🚀🔧🔥

**You said:**

Unknown Resource NULL NULL NULL 1171 Unknown Resource NULL NULL NULL 1172 Unknown Resource NULL NULL NULL 1177 Unknown Resource NULL NULL NULL 1183 Unknown Resource NULL NULL NULL 1184 Unknown Resource NULL NULL NULL 1185 Unknown Resource NULL NULL NULL 1187 Unknown Resource NULL NULL NULL 1188 Unknown Resource NULL NULL NULL 1189 Unknown Resource NULL NULL NULL 1200

**ChatGPT said:**

**Alright, mate!** ✅ The **resource IDs are there**, but **ResourceName, Role, MaxUnits, and CalendarID are showing as**NULL.

That means either:  
1️⃣ **The data didn’t exist in P6 (**RSRC**) for these fields.**  
2️⃣ **The**INSERT**statement didn’t pull the expected values.**

**🔹 Step 4: Check What’s Missing in P6 (**RSRC**)**

**✅ Run This Query**

sql

Copy

SELECT rsrc\_short\_name, rsrc\_name, role\_id, def\_qty\_per\_hr, clndr\_id FROM RSRC WHERE rsrc\_short\_name IN ('1171', '1172', '1177', '1183', '1184', '1185');

👉 **This checks if P6 actually has values for these resources.**  
👉 **If everything in P6 is also**NULL**, then the data doesn’t exist.**  
👉 **If P6 has values, then we need to fix the insert.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the results.**  
✅ **If P6 has data, we’ll fix the**INSERT**statement.**  
✅ **If P6 is missing data, we’ll decide how to handle**NULLs**.**

🚀 **Let me know what you find, mate! 🔧🔥**

**You said:**

that came up blank mate, ....those resources are not in rsrc

**ChatGPT said:**

**Got it, mate!** ✅ Since those resources **don’t exist in**RSRC, that means they were likely **already in**dbo.Resources**before we inserted P6 data.**

**🔹 Step 5: Check If These Were Pre-Existing**

Let’s confirm whether these **"Unknown Resource"** rows were already in dbo.Resources **before** we did the P6 insert.

**✅ Run This Query**

sql

Copy

SELECT \* FROM dbo.Resources WHERE ResourceName IS NULL;

👉 **If these were there before, we just need to clean them up.**  
👉 **If they were inserted from P6, we’ll fix the mapping.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the results.**  
✅ **If these were pre-existing, we’ll clean them up.**  
✅ **If they came from P6, we’ll find out why.**

🚀 **Let’s get this sorted, mate! 🔧🔥**

**You said:**

that came up blank as well, just the headers.

**ChatGPT said:**

**Alright, mate!** ✅ Since dbo.Resources **had no pre-existing "Unknown Resource" rows**, that means these **must have been inserted from P6**—but P6 didn't have corresponding details.

**🔹 Why Did This Happen?**

* The INSERT query **only pulled data that existed in**RSRC.
* If **some**ResourceID**s were referenced elsewhere (like**ResourceAssignments**) but not in**RSRC, they got inserted **without details** (hence the NULLs).

**🔹 Step 6: Find Where These "Unknown Resources" Came From**

Let's check **if these resources exist in**ResourceAssignments**but not**RSRC.

**✅ Run This Query**

sql

Copy

SELECT DISTINCT RA.ResourceID FROM dbo.ResourceAssignments RA LEFT JOIN RSRC R ON RA.ResourceID = R.rsrc\_short\_name WHERE R.rsrc\_short\_name IS NULL;

👉 **If this returns results, it means these resources exist in**ResourceAssignments**but not in P6's**RSRC**table.**  
👉 **If it’s blank, we need to check further.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the results.**  
✅ **If these resources exist in**ResourceAssignments**but not**RSRC**, we’ll decide how to handle them.**  
✅ **If it’s blank, we’ll dig deeper.**

🚀 **Let me know what you find, mate! 🔧🔥**

**You said:**

6980 7055 1177 6947 4454 6684 6911 6996 6901 6604 6878 6993 7022 7027 4484 7046 6601 6967 6952 6973 7058 6926 6985 6598 6915 1187 1205 6992 7036 6593 6923 6905 1189 1533 6948 6936 6968 6933 4458 6976 7052 6693 7057 7016 6686 6981 7006 7028 7026 6902 7048 6998 7011 4473 6602 6995 7053 6592 6987 6953 6961 6978 6683 6912 7043 7002 6742 7059 6960 6946 6928 6966 7008 4448 4449 6917 7012 6984 4460 7038 6907 6940 7056 4483 1837 6938 7018 6904 4462 4459 6609 4478 4455 6935 6736 7023 6925 4423 6949 7050 6914 6969 7031 6903 1183 7032 6941 7001 7042 6691 6982 6583 6913 6955 6582 6695 1200 6994 6999 7035 7049 6990 6942 6596 6697 6962 6974 6979 4482 7020 6591 6975 4481 4453 6688 7003 7040 6929 6924 6957 4479 6930 6920 7033 7021 7005 7054 6689 6600 6908 7030 4472 7041 6934 6939 6608 7000 6971 4475 6918 6983 1203 7013 6590 6585 6970 7029 7010 6692 6650 6958 1172 6959 6954 7051 7024 6587 6997 4476 6927 6932 6690 6687 7044 6986 6581 6951 7045 7007 6931 4447 7034 6910 6916 7039 6900 6595 7004 6685 6988 7017 6989 6743 6937 6965 6991 6956 6909 7014 7025 4461 1171 6580 6977 7019 4450 6607 7015 6906 6972 6944 7009 6594 6649 6919 6945 6605 6964 1188 6950 6694 6921 6922 7037 6588 1185 6943 1184 7047 6963 4452 6696 we need the corresponding resource\_short\_name, to each of these codes, as those are the real P6 resource\_ids

**ChatGPT said:**

**Got it, mate!** ✅ These are **P6**ResourceIDs**from**ResourceAssignments, but we need to **find their corresponding**resource\_short\_name**from**RSRC.

**🔹 Step 7: Get**resource\_short\_name**for These**ResourceIDs

**✅ Run This Query**

sql

Copy

SELECT rsrc\_id, rsrc\_short\_name FROM RSRC WHERE rsrc\_id IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **This will return the**resource\_short\_name**for each**ResourceID**in P6.**  
👉 **Any missing values are resources that don’t exist in**RSRC**.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the results.**  
✅ **If any**resource\_short\_name**values are missing, we’ll handle them properly.**

🚀 **Let’s get this sorted, mate! 🔧🔥**

**You said:**

bingo mate, thats the code I want = SELECT rsrc\_id, rsrc\_short\_name FROM RSRC WHERE rsrc\_id IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

**ChatGPT said:**

**Bingo, mate!** ✅ Now you’ve got the exact P6 resource\_short\_name values for the missing ResourceIDs! 🎯🔥

**🔹 What’s Next?**

Now that we have the correct mapping, do you want to:  
1️⃣ **Update**dbo.Resources**with the correct**ResourceID**values from P6?**  
2️⃣ **Use this data to generate a report?**  
3️⃣ **Export it to Excel for further analysis?**

Let me know how you want to proceed, mate! 🚀🔧🔥

**You said:**

yes, we need to map resource\_short\_name into the dbo.Resources table,

**ChatGPT said:**

**Got it, mate!** ✅ Now we need to **update**dbo.Resources by mapping resource\_short\_name from RSRC based on ResourceID.

**🔹 Step 8: Update**dbo.Resources**with**resource\_short\_name

Now we’ll **match the missing**ResourceID**s in**dbo.Resources**to**RSRC.resource\_short\_name.

**✅ Run This Query**

sql

Copy

UPDATE R SET R.ResourceID = RSR.resource\_short\_name FROM dbo.Resources R JOIN RSRC RSR ON R.ResourceID = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **This updates**dbo.Resources**so that the correct**resource\_short\_name**is used instead of**ResourceID**.**  
👉 **Now all resources will match their correct names from P6.**

**🔹 What You Need to Do Next**

✅ **Run the query above and confirm how many rows were updated.**  
✅ **If you get an error, send me the message, and we’ll fix it.**  
✅ **Once done, we’ll verify the final data.**

🚀 **Let me know how it goes, mate! 🔧🔥**

**You said:**

we need to add RSR.resource\_short\_name to dbo.Resources first?

**ChatGPT said:**

**Yes, mate!** ✅ Before we can update ResourceID with resource\_short\_name, we **need to first add**resource\_short\_name**as a new column in**dbo.Resources so we don't overwrite anything important.

**🔹 Step 8.1: Add**resource\_short\_name**to**dbo.Resources

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ADD ResourceShortName NVARCHAR(50) NULL;

👉 **This creates a new column (**ResourceShortName**) where we will store**resource\_short\_name**from P6.**

**🔹 Step 8.2: Populate**ResourceShortName**from P6**

**✅ Run This Query**

sql

Copy

UPDATE R SET R.ResourceShortName = RSR.resource\_short\_name FROM dbo.Resources R JOIN RSRC RSR ON R.ResourceID = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **Now, we store**resource\_short\_name**in a new column (**ResourceShortName**).**  
👉 **This keeps**ResourceID**as it is while adding a human-readable name.**

**🔹 Step 8.3: Verify the Update**

**✅ Run This Query**

sql

Copy

SELECT TOP 10 ResourceID, ResourceShortName FROM dbo.Resources;

👉 **Check that**ResourceShortName**now contains the correct names from P6.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.1 to add the column.**  
✅ **Run Step 8.2 to update it with P6 data.**  
✅ **Run Step 8.3 to verify the update.**  
✅ **Let me know if it worked or if any errors came up!**

🚀 **We’re nearly there, mate! 🔧🔥**

**You said:**

we need to verify the column exists, before we add the data....what is the code for verifying columns

**ChatGPT said:**

**Good thinking, mate!** ✅ Before adding the ResourceShortName column, let’s **check if it already exists in**dbo.Resources to avoid errors.

**🔹 Step 8.1: Check If**ResourceShortName**Exists**

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND COLUMN\_NAME = 'ResourceShortName';

👉 **If this returns a row, the column already exists.**  
👉 **If it returns blank, we need to add it.**

**🔹 Step 8.2: Add**ResourceShortName**(If It’s Missing)**

**✅ Run This Query**

sql

Copy

IF NOT EXISTS ( SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND COLUMN\_NAME = 'ResourceShortName' ) BEGIN ALTER TABLE dbo.Resources ADD ResourceShortName NVARCHAR(50) NULL; END;

👉 **This only adds**ResourceShortName**if it doesn’t already exist.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.1 to check if the column exists.**  
✅ **If it doesn’t exist, run Step 8.2 to add it.**  
✅ **Let me know if the column is already there or if it had to be added!**

🚀 **Once that’s sorted, we’ll update it with P6 data! 🔧🔥**

**You said:**

it needs to be resource\_short\_name, right, as per the details in the other table

**ChatGPT said:**

**Yes, mate!** ✅ The column name should match exactly with P6’s RSRC table, so we need to use resource\_short\_name instead of ResourceShortName.

**🔹 Step 8.1: Check If**resource\_short\_name**Exists in**dbo.Resources

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND COLUMN\_NAME = 'resource\_short\_name';

👉 **If this returns a row, the column already exists.**  
👉 **If it’s blank, we need to add the column.**

**🔹 Step 8.2: Add**resource\_short\_name**(If It’s Missing)**

**✅ Run This Query**

sql

Copy

IF NOT EXISTS ( SELECT 1 FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND COLUMN\_NAME = 'resource\_short\_name' ) BEGIN ALTER TABLE dbo.Resources ADD resource\_short\_name NVARCHAR(50) NULL; END;

👉 **This ensures**resource\_short\_name**is only added if it doesn’t already exist.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.1 to check if**resource\_short\_name**exists.**  
✅ **If it’s missing, run Step 8.2 to add it.**  
✅ **Let me know if it was already there or had to be added!**

🚀 **Once that’s done, we’ll move to updating it with P6 data! 🔧🔥**

**You said:**

right, the heading is there, so how do we add the p6 data, into resource\_short\_name, where are we taking it from

**ChatGPT said:**

**Perfect, mate!** ✅ Since resource\_short\_name already exists in dbo.Resources, we now need to **populate it with the correct P6 data** from RSRC.

**🔹 Step 8.3: Update**resource\_short\_name**with P6 Data**

Now, we **map**resource\_short\_name**from P6’s**RSRC**table into**dbo.Resources.

**✅ Run This Query**

sql

Copy

UPDATE R SET R.resource\_short\_name = RSR.resource\_short\_name FROM dbo.Resources R JOIN RSRC RSR ON R.ResourceID = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **This updates**resource\_short\_name**in**dbo.Resources**using P6’s**RSRC.resource\_short\_name  
👉 **Now, every resource will have a human-readable identifier instead of just a numeric ID!**

**🔹 Step 8.4: Verify the Update**

**✅ Run This Query**

sql

Copy

SELECT TOP 10 ResourceID, resource\_short\_name FROM dbo.Resources;

👉 **This checks if the update worked and that the**resource\_short\_name**values are now filled.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.3 to update**resource\_short\_name**from P6.**  
✅ **Run Step 8.4 to verify the update.**  
✅ **Let me know how many rows were updated and if everything looks good!**

🚀 **Once this is done, we can move on to reporting! 🔧🔥**

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 2 Invalid column name 'resource\_short\_name'. Total execution time: 00:00:00.004

**ChatGPT said:**

**Alright, mate!** ✅ The error **“Invalid column name ‘resource\_short\_name’”** means that the column **does not exist in**dbo.Resources, or the name is slightly different.

**🔹 Step 8.5: Double-Check Column Names in**dbo.Resources

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources';

👉 **This will show the exact column names in**dbo.Resources**.**  
👉 **Check if**resource\_short\_name**is there or if it's named something slightly different.**

**🔹 What You Need to Do Next**

✅ **Run the query above and send me the exact column names.**  
✅ **Once we confirm the correct column name, we’ll adjust the update query.**

🚀 **Let’s fix this and get it right! 🔧🔥**

**You said:**

ResourceName Role MaxUnits CalendarID ResourceID ResourceShortName resource\_short\_name

**ChatGPT said:**

**Nice one, mate!** ✅ You’ve got **both**ResourceShortName**and**resource\_short\_name in dbo.Resources. That’s likely causing the issue—SQL Server is case-insensitive for column names, but having **two similar names** might be confusing the query.

**🔹 Step 8.6: Remove Duplicate Column (If Needed)**

Since ResourceShortName and resource\_short\_name are both there, we need to remove one of them **before** updating the data.

**✅ Run This Query to Identify Which One to Keep**

sql

Copy

SELECT TOP 10 ResourceShortName, resource\_short\_name FROM dbo.Resources;

👉 **If one column is already filled and the other is empty, we drop the empty one.**  
👉 **If both are empty, we’ll pick**resource\_short\_name**(since that’s the P6 column name).**

**🔹 Step 8.7: Drop the Unnecessary Column**

If ResourceShortName is empty and resource\_short\_name is what we want to use, **drop**ResourceShortName:

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP COLUMN ResourceShortName;

👉 **This keeps**resource\_short\_name**and removes the duplicate.**

**🔹 Step 8.8: Update**resource\_short\_name**from P6**

Now that we have just **one correct column**, let’s update it:

**✅ Run This Query**

sql

Copy

UPDATE R SET R.resource\_short\_name = RSR.resource\_short\_name FROM dbo.Resources R JOIN RSRC RSR ON R.ResourceID = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **Now that we’re using the correct column (**resource\_short\_name**), this update should work!**  
👉 **Each resource in**dbo.Resources**will now have the correct**resource\_short\_name**from P6.**

**🔹 Final Steps**

✅ **Run Step 8.6 to check if**ResourceShortName**and**resource\_short\_name**both exist.**  
✅ **If needed, run Step 8.7 to remove**ResourceShortName**.**  
✅ **Run Step 8.8 to update**resource\_short\_name**from P6.**  
✅ **Finally, verify the update with:**

sql

Copy

SELECT TOP 10 ResourceID, resource\_short\_name FROM dbo.Resources;

**🔹 What You Need to Do Next**

✅ **Run the steps in order and let me know if it worked!**  
✅ **Once confirmed, we’ll move to reporting!**

🚀 **We’re nearly done, mate! 🔧🔥**

**You said:**

they are both NULL, will that allow us to remove the column? ResourceShortName

**ChatGPT said:**

**Yes, mate!** ✅ Since **both**ResourceShortName**and**resource\_short\_name**are NULL**, we can safely **drop**ResourceShortName without losing any data.

**🔹 Step 8.7: Drop**ResourceShortName**(If It’s Empty)**

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources DROP COLUMN ResourceShortName;

👉 **This removes**ResourceShortName**and keeps**resource\_short\_name**.**  
👉 **No data is lost since both columns are empty.**

**🔹 Step 8.8: Update**resource\_short\_name**from P6**

**✅ Run This Query**

sql

Copy

UPDATE R SET R.resource\_short\_name = RSR.resource\_short\_name FROM dbo.Resources R JOIN RSRC RSR ON R.ResourceID = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **Now**resource\_short\_name**will be correctly updated from P6!**

**🔹 Step 8.9: Verify the Update**

**✅ Run This Query**

sql

Copy

SELECT TOP 10 ResourceID, resource\_short\_name FROM dbo.Resources;

👉 **Check that the**resource\_short\_name**values are now filled.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.7 to drop**ResourceShortName**.**  
✅ **Run Step 8.8 to update**resource\_short\_name**from P6.**  
✅ **Run Step 8.9 to verify the update.**  
✅ **Let me know how many rows were updated and if everything looks good!**

🚀 **We’re nearly there, mate! 🔧🔥**

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 2 Invalid column name 'resource\_short\_name'. Total execution time: 00:00:00.005, why is this always coming back with invalid column name, ...we have proved its there....

**ChatGPT said:**

**I hear you, mate!** ✅ We **confirmed** that resource\_short\_name exists, but SQL **still throws "Invalid column name"**—which means something is **off in referencing it**.

**🔹 Step 8.10: Triple-Check Column Name (Exact Spelling & Schema)**

Let’s **check the exact schema, table, and column name** to see if something’s off.

**✅ Run This Query**

sql

Copy

SELECT TABLE\_SCHEMA, COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources';

👉 **This will confirm:**  
✅ The **exact spelling** of resource\_short\_name.  
✅ The **schema name** (if it’s dbo.Resources or something else).  
✅ The **data type** to ensure no conflicts exist.

**🔹 Step 8.11: Check If**resource\_short\_name**Is Actually Usable**

Next, let’s **run a simple SELECT to confirm SQL can retrieve it**:

**✅ Run This Query**

sql

Copy

SELECT TOP 10 resource\_short\_name FROM dbo.Resources;

👉 **If this query fails, SQL still doesn’t recognize the column.**  
👉 **If it works, then the issue is in our UPDATE query’s syntax.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.10 to confirm exact column details.**  
✅ **Run Step 8.11 to test if SQL can actually fetch data from it.**  
✅ **Let me know what happens—this will tell us exactly where the issue is!**

🚀 **Once we crack this, we’re golden! 🔧🔥**

**You said:**

dbo ResourceName nvarchar dbo Role nvarchar dbo MaxUnits int dbo CalendarID int dbo ResourceID nvarchar dbo resource\_short\_name nvarchar, ok so we have those here, .... is it a problem with where its coming from?

**ChatGPT said:**

**Great, mate!** ✅ We’ve confirmed that resource\_short\_name**exists in**dbo.Resources, and it's an NVARCHAR type—so the issue is likely in **how we’re referencing it in the update query.**

**🔹 Step 8.12: Check If**resource\_short\_name**Can Be Selected from**RSRC

Since SQL says **“Invalid column name”**, the issue could be:

* 🚨 **It doesn’t exist in**RSRC (or is named slightly differently).
* 🚨 **A typo in the table reference** (RSRC vs. dbo.RSRC).

**✅ Run This Query**

sql

Copy

SELECT TOP 10 rsrc\_id, resource\_short\_name FROM RSRC;

👉 **If this fails,**resource\_short\_name**is named something else in**RSRC**.**  
👉 **If it works, then our update query just needs fixing.**

**🔹 Step 8.13: Get the Exact Column Names from**RSRC

If Step 8.12 **fails**, we need to find the actual column name in RSRC.

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'RSRC';

👉 **This will return the correct column names in**RSRC**.**  
👉 **Check if**resource\_short\_name**is there or if it’s named differently (e.g.,**rsrc\_short\_name**).**

**🔹 What You Need to Do Next**

✅ **Run Step 8.12 to check if**resource\_short\_name**exists in**RSRC**.**  
✅ **If Step 8.12 fails, run Step 8.13 to get the exact column name.**  
✅ **Let me know the results, and we’ll adjust the update query accordingly!**

🚀 **Once we fix this, we can finally update the data! 🔧🔥**

**You said:**

Started executing query at Line 1 Msg 207, Level 16, State 1, Line 1 Invalid column name 'resource\_short\_name'. Total execution time: 00:00:00.002, we looked at the columns we had to choose from, and it was there...so why is it now saying its not.

**ChatGPT said:**

**I get your frustration, mate!** ✅ We've **confirmed** that resource\_short\_name is in dbo.Resources, but SQL **still says it's invalid**—which likely means:

🚨 **We're querying the wrong table (**RSRC**instead of**dbo.RSRC**)**  
🚨 **The column name in**RSRC**is actually different (maybe**rsrc\_short\_name**)**  
🚨 **The table reference is missing its schema (**dbo.RSRC**vs.**RSRC**)**

**🔹 Step 8.14: Check the Exact Column Name in**RSRC

Let’s **double-check** what RSRC calls this column.

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'RSRC';

👉 **Look at the exact column names SQL returns.**  
👉 **Check if it’s actually**rsrc\_short\_name**,**rsrc\_name**, or something else.**

**🔹 Step 8.15: Select the Column from**RSRC

Once we confirm the correct name, let’s **see if SQL lets us query it**.

**✅ Run This Query**

sql

Copy

SELECT TOP 10 rsrc\_id, rsrc\_short\_name FROM RSRC;

OR (if RSRC is in dbo schema)

sql

Copy

SELECT TOP 10 rsrc\_id, rsrc\_short\_name FROM dbo.RSRC;

👉 **If this fails, the column is named something different.**  
👉 **If it works, then our update query just needs fixing.**

**🔹 What You Need to Do Next**

✅ **Run Step 8.14 to check column names in**RSRC**.**  
✅ **Run Step 8.15 to test if SQL lets you select the column.**  
✅ **Let me know what the exact column name is, and we’ll fix the update query!**

🚀 **We’re so close to cracking this, mate! 🔧🔥**

**You said:**

rsrc\_id clndr\_id rsrc\_seq\_num timesheet\_flag active\_flag rsrc\_type auto\_compute\_act\_flag ot\_flag def\_cost\_qty\_link\_flag rsrc\_short\_name rsrc\_name parent\_rsrc\_id xfer\_complete\_day\_cnt xfer\_notstart\_day\_cnt ts\_approve\_user\_id user\_id role\_id curr\_id last\_checksum shift\_id ot\_factor def\_qty\_per\_hr guid cost\_qty\_type office\_phone other\_phone employee\_code rsrc\_title\_name email\_addr unit\_id pobs\_id rsrc\_notes intg\_type update\_date location\_id update\_user create\_date create\_user delete\_session\_id delete\_date rsrc\_id clndr\_id rsrc\_seq\_num timesheet\_flag active\_flag rsrc\_type auto\_compute\_act\_flag ot\_flag def\_cost\_qty\_link\_flag rsrc\_short\_name rsrc\_name parent\_rsrc\_id xfer\_complete\_day\_cnt xfer\_notstart\_day\_cnt ts\_approve\_user\_id user\_id role\_id curr\_id last\_checksum shift\_id ot\_factor def\_qty\_per\_hr guid cost\_qty\_type office\_phone other\_phone employee\_code rsrc\_title\_name email\_addr unit\_id pobs\_id rsrc\_notes intg\_type update\_date location\_id update\_user create\_date create\_user delete\_session\_id delete\_date, its column 50...so its there

**ChatGPT said:**

**Good news, mate!** ✅ The column rsrc\_short\_name **is definitely in**RSRC, so now we just need to **correct the update query** to use the right column name.

**🔹 Step 8.16: Update**resource\_short\_name**in**dbo.Resources

Since the correct column name is rsrc\_short\_name (not resource\_short\_name), we **must update our query** to reflect that.

**✅ Run This Query**

sql

Copy

UPDATE R SET R.resource\_short\_name = RSR.rsrc\_short\_name FROM dbo.Resources R JOIN dbo.RSRC RSR ON R.ResourceID = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **Key Fixes:**  
✅ **Changed**resource\_short\_name**to**rsrc\_short\_name (the correct column name).  
✅ **Used**dbo.RSRC**to ensure SQL finds the table correctly.**  
✅ **Casting**rsrc\_id**to**NVARCHAR(50)**to match**ResourceID**in**dbo.Resources**.**

**🔹 Step 8.17: Verify the Update**

**✅ Run This Query**

sql

Copy

SELECT TOP 10 ResourceID, resource\_short\_name FROM dbo.Resources;

👉 **Check if**resource\_short\_name**is now populated!**

**🔹 What You Need to Do Next**

✅ **Run Step 8.16 to update**resource\_short\_name**from**rsrc\_short\_name**.**  
✅ **Run Step 8.17 to verify the update.**  
✅ **Let me know how many rows were updated and if everything looks good!**

🚀 **This should finally crack it! 🔧🔥**

**You said:**

its saying that resource\_short\_name in dbo.Resources is still a int, and not an nvarchar(50)...but I thought we sorted that.

**ChatGPT said:**

**Alright, mate!** ✅ If resource\_short\_name**is still an**INT, then SQL didn’t apply the change properly. We need to **double-check its data type and fix it.**

**🔹 Step 8.18: Verify Data Type in**dbo.Resources

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'Resources' AND COLUMN\_NAME = 'resource\_short\_name';

👉 **This will confirm if**resource\_short\_name**is still**INT**instead of**NVARCHAR(50)**.**  
👉 **If it's**INT**, we need to manually change it.**

**🔹 Step 8.19: Convert**resource\_short\_name**to**NVARCHAR(50)

If **Step 8.18 confirms it’s**INT, we need to **ALTER the column**.

**✅ Run This Query**

sql

Copy

ALTER TABLE dbo.Resources ALTER COLUMN resource\_short\_name NVARCHAR(50);

👉 **This ensures**resource\_short\_name**can store text values from P6.**

**🔹 Step 8.20: Try Updating Again**

Once the column is fixed, **re-run the update query from Step 8.16** to populate it with P6 data.

**🔹 What You Need to Do Next**

✅ **Run Step 8.18 to check the current data type.**  
✅ **If**resource\_short\_name**is**INT**, run Step 8.19 to change it to**NVARCHAR(50)**.**  
✅ **Once that’s done, re-run Step 8.16 to update the column with P6 data.**  
✅ **Finally, verify the update with Step 8.17.**

🚀 **We’re gonna get this done right, mate! 🔧🔥**

**You said:**

resource\_short\_name nvarchar 50, ..hmm this is confusing, I got this an error, that it was an int,

**ChatGPT said:**

**Yeah, mate, that’s strange!** ✅ The query says resource\_short\_name**is**NVARCHAR(50), but SQL still **throws an error saying it's an**INT—which means one of the following is happening:

**🚨 Possible Reasons for the Error:**

1️⃣ **There might be an old cached execution plan**—SQL is still treating it as INT.  
2️⃣ **The column is referenced elsewhere** as an INT, and that’s causing a conflict.  
3️⃣ **The error might be coming from**RSRC.rsrc\_id**instead**—not resource\_short\_name.

**🔹 Step 8.21: Refresh SQL Server’s Cache**

First, let’s **clear SQL’s cache** to make sure it recognizes the change.

**✅ Run This Query**

sql

Copy

DBCC FREEPROCCACHE;

👉 **This clears execution plans, forcing SQL to re-evaluate table structures.**  
👉 **Now try running the update query again.**

**🔹 Step 8.22: Manually Convert**rsrc\_id**to**NVARCHAR(50)

If the update query still **throws the same error**, the problem might actually be RSRC.rsrc\_id instead of resource\_short\_name.

Let’s **confirm the data type of**rsrc\_id**in**RSRC:

**✅ Run This Query**

sql

Copy

SELECT COLUMN\_NAME, DATA\_TYPE FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'RSRC' AND COLUMN\_NAME = 'rsrc\_id';

👉 **If**rsrc\_id**is**INT**, we need to CAST it to**NVARCHAR(50)**in the update query.**

**🔹 Step 8.23: Modify Update Query (If Needed)**

If rsrc\_id**is an**INT, update the query like this:

**✅ Run This Query**

sql

Copy

UPDATE R SET R.resource\_short\_name = RSR.rsrc\_short\_name FROM dbo.Resources R JOIN dbo.RSRC RSR ON CAST(R.ResourceID AS NVARCHAR(50)) = CAST(RSR.rsrc\_id AS NVARCHAR(50)) WHERE R.ResourceID IN ( 6980, 7055, 1177, 6947, 4454, 6684, 6911, 6996, 6901, 6604, 6878, 6993, 7022, 7027, 4484, 7046, 6601, 6967, 6952, 6973, 7058, 6926, 6985, 6598, 6915, 1187, 1205, 6992, 7036, 6593, 6923, 6905, 1189, 1533, 6948, 6936, 6968, 6933, 4458, 6976, 7052, 6693, 7057, 7016, 6686, 6981, 7006, 7028, 7026, 6902, 7048, 6998, 7011, 4473, 6602, 6995, 7053, 6592, 6987, 6953, 6961, 6978, 6683, 6912, 7043, 7002, 6742, 7059, 6960, 6946, 6928, 6966, 7008, 4448, 4449, 6917, 7012, 6984, 4460, 7038, 6907, 6940, 7056, 4483, 1837, 6938, 7018, 6904, 4462, 4459, 6609, 4478, 4455, 6935, 6736, 7023, 6925, 4423, 6949, 7050, 6914, 6969, 7031, 6903, 1183, 7032, 6941, 7001, 7042, 6691, 6982, 6583, 6913, 6955, 6582, 6695, 1200, 6994, 6999, 7035, 7049, 6990, 6942, 6596, 6697, 6962, 6974, 6979, 4482, 7020, 6591, 6975, 4481, 4453, 6688, 7003, 7040, 6929, 6924, 6957, 4479, 6930, 6920, 7033, 7021, 7005, 7054, 6689, 6600, 6908, 7030, 4472, 7041, 6934, 6939, 6608, 7000, 6971, 4475, 6918, 6983, 1203, 7013, 6590, 6585, 6970, 7029, 7010, 6692, 6650, 6958, 1172, 6959, 6954, 7051, 7024, 6587, 6997, 4476, 6927, 6932, 6690, 6687, 7044, 6986, 6581, 6951, 7045, 7007, 6931, 4447, 7034, 6910, 6916, 7039, 6900, 6595, 7004, 6685, 6988, 7017, 6989, 6743, 6937, 6965, 6991, 6956, 6909, 7014, 7025, 4461, 1171, 6580, 6977, 7019, 4450, 6607, 7015, 6906, 6972, 6944, 7009, 6594, 6649, 6919, 6945, 6605, 6964, 1188, 6950, 6694, 6921, 6922, 7037, 6588, 1185, 6943, 1184, 7047, 6963, 4452, 6696 );

👉 **Key Fixes:**  
✅ **Forced both**ResourceID**and**rsrc\_id**to**NVARCHAR(50)**.**  
✅ **Cleared cache before running the query.**  
✅ **Ensured table reference is correct (**dbo.RSRC**).**

**🔹 What You Need to Do Next**

✅ **Run Step 8.21 to clear SQL’s cache.**  
✅ **Run Step 8.22 to check**rsrc\_id**data type.**  
✅ **If**rsrc\_id**is an**INT**, run Step 8.23 to modify the update query.**  
✅ **Let me know what happens!**

🚀 **We’ll smash this issue, mate! 🔧🔥**

**You said:**

rsrc\_id, is an INT on both tables, but surely we are only using it as a reference point, to bring in the resource\_short\_name which is an nvarchar(50)...so yes, change the cast formula, to use rsrc\_id as an int.

**ChatGPT said:**

​

You previously reached your usage cap for GPT-4, but you can now try sending your message again

Retry