Detailed Instructions for ChatGPT Data Professional Tasks

Beginner Level Tasks

Task 1: Data Cleaning Assistant

Key Terms:

- Data quality issues: Problems in data that affect its accuracy and reliability
- Data validation: Process of ensuring data meets specific quality standards
- Data cleaning: Process of fixing or removing incorrect, corrupted, or irrelevant data

Step-by-Step Instructions:

1. Copy the provided messy dataset exactly as shown
2. Ask ChatGPT: "Can you help me identify all data quality issues in this dataset?"
3. For each issue identified, ask: "Why is this considered a data quality problem?"
4. Request Python code: "Please write Python code to clean this dataset and handle all the issues you identified"
5. Ask: "What data validation rules would you recommend implementing to prevent these issues in the future?"

Expected Outcomes:

- Understanding of inconsistent date formats
- Handling of missing values (null)
- Standardization of region names
- Proper formatting of currency values

Task 2: SQL Query Optimization

Key Terms:

- Query optimization: Process of improving SQL query performance
- Indexing: Database structure that speeds up data retrieval
- JOIN operations: Combining rows from different tables

Step-by-Step Instructions:

1. Share the provided SQL query
2. Ask ChatGPT: "What are the potential performance issues in this query?"
3. Request optimization: "How can we optimize this query for better performance?"
4. Ask about indexes: "What indexes would you recommend for this query and why?"
5. Request explanation: "Please explain how each suggested improvement helps performance"

Expected Outcomes:

- Column selection optimization
- Index recommendations
- Join optimization strategies
- Query restructuring suggestions

Task 3: Data Pipeline Debug

Key Terms:

- ETL: Extract, Transform, Load
- Data pipeline: Series of data processing steps
- Error handling: Managing and responding to errors in code

Step-by-Step Instructions:

1. Share the provided Python code
2. Ask: "What potential issues do you see in this code?"
3. Request error handling: "How can we add proper error handling to this code?"
4. Ask for improvements: "What best practices could we implement here?"
5. Request complete solution: "Can you provide an improved version with all suggestions implemented?"

Expected Outcomes:

- Error handling implementation
- Code efficiency improvements

- Best practices implementation
- Documentation suggestions

Intermediate Level Tasks

Task 4: Feature Engineering

Key Terms:

- Feature engineering: Process of creating new features from existing data
- Feature scaling: Normalizing the range of features
- Customer churn: When customers stop using a service

Step-by-Step Instructions:

1. Share the dataset columns
2. Ask: "What relevant features can we create from these columns for churn prediction?"
3. Request code: "Please generate Python code for creating these features"
4. Ask for explanation: "Why would each suggested feature be useful for predicting churn?"
5. Request scaling guidance: "What scaling methods would work best for these features?"

Expected Outcomes:

- Time-based features
- Behavioral indicators
- Statistical aggregations
- Scaling recommendations

Task 5: Data Architecture Design

Key Terms:

- Dimensional model: Data warehouse modeling technique
- Slowly changing dimensions: Tracking historical changes
- Partitioning: Dividing tables into smaller, manageable parts

Step-by-Step Instructions:

1. Present the e-commerce scenario
2. Ask: "What would be the key fact and dimension tables needed?"
3. Request SQL: "Please provide CREATE TABLE statements for the core tables"
4. Ask about SCD: "How should we handle slowly changing dimensions for customer data?"
5. Request partitioning strategy: "What partitioning approach would you recommend?"

Expected Outcomes:

- Dimensional model diagram
- Table definitions
- SCD implementation strategy
- Partitioning recommendations

Task 6: Performance Monitoring

Key Terms:

- Metrics collection: Gathering performance data
- Error tracking: Monitoring and logging errors
- Alerting logic: Rules for sending notifications

Step-by-Step Instructions:

1. Share the basic monitoring code
2. Ask: "How can we add comprehensive performance metrics?"
3. Request error tracking: "What error tracking implementation would you suggest?"
4. Ask about alerts: "Can you provide code for implementing alerting logic?"
5. Request dashboard queries: "What queries would be useful for monitoring dashboards?"

Expected Outcomes:

- Enhanced monitoring code
- Error handling implementation

- Alerting system design
- Dashboard query examples

Advanced Level Tasks

Task 7: Machine Learning Pipeline

Key Terms:

- Imbalanced data: Dataset where classes are not equally represented
- Cross-validation: Technique for assessing ML model performance
- Preprocessing: Preparing data for machine learning

Step-by-Step Instructions:

1. Present the classification problem
2. Ask: "What preprocessing steps are needed for imbalanced data?"
3. Request model selection: "Which models would work best for this scenario?"
4. Ask for code: "Can you provide cross-validation code for this case?"
5. Request metrics: "What evaluation metrics should we use and why?"

Expected Outcomes:

- Preprocessing pipeline
- Model selection rationale
- Cross-validation implementation
- Evaluation framework

Task 8: Data Quality Framework

Key Terms:

- Test cases: Scenarios for testing data quality
- Quality framework: System for ensuring data quality
- Automated checks: Programmatic data validation

Step-by-Step Instructions:

1. Share quality check requirements
2. Ask: "What test cases should we implement?"
3. Request code: "Can you generate code for these quality checks?"
4. Ask about reporting: "How should we structure the quality reporting?"
5. Request implementation: "What's the best way to automate these checks?"

Expected Outcomes:

- Test case definitions
- Testing code implementation
- Reporting structure
- Automation strategy

Task 9: Real-time Analytics

Key Terms:

- Streaming data: Continuous flow of data
- Windowing functions: Time-based data aggregation
- Real-time processing: Immediate data analysis

Step-by-Step Instructions:

1. Present streaming scenario
2. Ask: "What architecture components do we need?"
3. Request processing logic: "How should we process this streaming data?"
4. Ask about aggregation: "What aggregation strategies would work best?"
5. Request implementation: "Can you provide windowing function examples?"

Expected Outcomes:

- Architecture diagram
- Processing logic code
- Aggregation strategies
- Window function examples

Task 10: Integration Challenge

Key Terms:

- End-to-end pipeline: Complete data processing system
- Architecture diagram: Visual representation of system design
- Deployment strategy: Plan for implementing the solution

Step-by-Step Instructions:

1. Present the transaction processing scenario
2. Ask: "Can you design an end-to-end architecture?"
3. Request implementation: "What would the key component code look like?"
4. Ask about monitoring: "How should we monitor this pipeline?"
5. Request deployment: "What deployment strategy would you recommend?"

Expected Outcomes:

- Complete architecture design
- Component implementations
- Monitoring strategy
- Deployment plan

General Tips for Using These Instructions:

1. Read through the entire task before starting
2. Look up any unfamiliar terms in the Key Terms section
3. Follow the step-by-step instructions in order
4. Save ChatGPT's responses for future reference
5. If stuck, try rephrasing the question using similar terms
6. Always verify and test any generated code
7. Ask for clarification if ChatGPT's response is unclear