**Student Name:** Michael Ayomide, FADELE

**Student Number:** 239032281

**Report Title:** Orchid Tech Innovations Ltd.'s Global Expansion: Technology and Big Data Challenges

**Course Code:** CETM 50

**Lecturer Name:** DR. Ashley Williamson

## 1.0 Introduction

Orchid Tech Innovations Ltd., a rising digital solutions and services company in the UK, inspires modern entrepreneurship. In a competitive market, this SME, founded a few years ago, has quickly established itself. The company specialises in digital marketing, analytics, and customer relationship management. A diversified customer base and strong development have resulted from Orchid Tech's agility and inventive approach to digital solutions.

Several Orchid Tech business areas work together to provide a complete service. In today's internet-driven industry, the company's digital marketing division works to boost brand awareness and consumer interaction (Kotler and Keller, 2021). One other important section, analytics, helps clients make informed decisions by interpreting data patterns (Davenport, 2014). Furthermore, their customer relationship management technology streamlines client interactions for a personalised and efficient service experience (Rogers, 2020).

With its rapid growth and expansion, Orchid Tech has faced various operational issues, mostly related to its IT infrastructure. Due to the company's siloed off-the-shelf IT solutions, data integration and efficiency have suffered. It's clear that a technical makeover is needed to improve operations and enable data sharing across corporate functions.

The purpose of this research is to critically assess Orchid Tech Innovations Ltd.'s expansion and consolidation of technological options. Considering the feasibility of integrating the company's IT systems will be a priority. It will also examine the usage of a Relational Database Management System (RDBMS) for Python-based Extract, Transform, Load (ETL) processes, which Vassiliadis (2019) says are essential to data consolidation.

With an emphasis on opening a Tokyo office, the research will also examine Orchid Tech's big data difficulties as it expands abroad. The regulatory, legal, and technological challenges related to managing and processing non-UK customer data will be examined. Scaling up corporate operations involves handling higher data volumes, velocity, and variety, as Laney (2021) states.

Methodically organised, the report covers these elements. After this introduction, Section 2 details the company's IT architecture and capacities. After assessing technology options, Section 3 makes a corporate recommendation. Relational databases for Python ETL workloads are criticised in Section 4, highlighting their pros and cons. In Section 5, legal, regulatory, and technological big data issues of international expansion are examined. The report finishes with a summary and Orchid Tech Innovations Ltd. action recommendations.

## 2.0 Background

Orchid Tech Innovations Ltd. represents 21st-century digital firms' lively and continuously changing landscape. Orchid Tech, a digital solution, and services SME has grown rapidly and gained clients. However, growth requires managing a more complicated IT infrastructure. Orchid Tech's IT architecture includes off-the-shelf software solutions for finance, HR, and

customer interactions.
Themistocleous and Chen (2022) found that start-ups' segmented IT approaches often contribute to operational inefficiencies as they grow. Orchid Tech's siloed systems demonstrate this issue. Fragmented systems isolate vital data, hampering cross-functional data integration and slowing data processing and decision-making (Fink and Neumann, 2017). Our walled data architecture hinders our capacity to comprehend our operations, which is essential for strategic planning (Peppard and Ward, 2016).
Fragmented IT infrastructure has major consequences. Luftman (2023) notes that such a system might lead to data management discrepancies and errors across an organisation. Orchid Tech's development complicates matters. Digital marketing, especially social media, has driven the company's growth (Kotler and Keller, 2021). To manage the complexity of digital engagement plans, this expansion requires more advanced IT solutions.
Orchid Tech has extensive geographic expansion aspirations. In-depth market research supports the company's choice to open a new office in Tokyo, Japan, a hub for both the local Japanese and East Asian markets (Ghemawat, 2021). Operations are more complicated due to foreign expansion. According to Mithas et al., (2023), coordinating UK and Tokyo office activities will require a more strong and unified IT infrastructure to support data exchange and efficient communication.

Orchid Tech must go from a segmented IT structure to an integrated one. This transition is necessary to maintain the company's growth and handle international expansion. This IT upgrade must address data integration and efficiency and position Orchid Tech to adapt to the digital economy.

## 3. Technology Solutions for Expansion

### 3.1 Technology Infrastructure Assessment

Orchid Tech Innovations Ltd.'s IT infrastructure is a mix of off-the-shelf solutions for certain company operations. This approach is initially cost-effective but presents issues as the company grows. Themistocleous and Chen (2022) says that a fragmented IT landscape causes operational inefficiencies, especially for rising firms. Orchid Tech's siloed systems prevent business function integration.

Luftman (2023) notes that this walled strategy slows data processing and decision-making. Manual data integration is needed since critical information is locked in individual systems. Manual intervention takes time and is prone to error, reducing data accuracy and reliability (Fink and Neumann, 2017). The inability to easily communicate and analyse data across the organisation inhibits Orchid Tech's strategic planning and market response (Peppard and Ward, 2016).

### 3.2 Potential Technology Solutions

More advanced technology is needed as Orchid Tech expands. This may be solved via iPaaS. iPaaS cloud services connect enterprise applications, systems, and data across environments (Dunie et al., 2015). This would solve data silos by integrating and flowing data between systems.

Cloud-based Enterprise Resource Planning (ERP) systems are another option. Cloud-based ERP solutions are perfect for fast-growing companies like Orchid Tech due to their scalability, flexibility, and cost-effectiveness, according to Mithas et al., (2023). Integration of business processes into a single system streamlines operations and improves data consistency.

Customer Relationship Management (CRM) solutions are other options. Kotler and Keller (2021) stress the relevance of CRM systems for customer interactions, data, and business operations. Orchid Tech could improve customer service and engagement with a powerful CRM solution that links consumer interactions across platforms.

### 3.3 Recommendation and Rationale

Orchid Tech should choose a cloud-based ERP system with a powerful CRM solution based on its needs and the analysed solutions. This would provide a platform for managing corporate processes, customer data, and operations across locations.

The benefits of this technique are many. First, a cloud-based ERP system is scalable and flexible, ideal for international expansion (Mithas et al., 2023). Secondly, incorporating a CRM system within the ERP framework centralises customer interactions and data and aligns them with other corporate activities, per Kotler and Keller (2021).
However, adopting and integrating these systems and training personnel to adapt to them may be difficult (Peppard and Ward, 2016). Cloud data storage and management raises security and compliance concerns (Dunie et al.,2015).

This recommendation fits Orchid Tech's growth and operational needs. For growth and market competitiveness, Orchid Tech may improve operational efficiency, data consistency, and customer relationship management by combining fragmented systems into a cloud-based ERP and CRM solution.

## 4. Relational database critique for Python ETL

### 4.1 Python ETL Process Overview

ETL methods are essential for data integration and administration, especially at Orchid Tech, which handles several data sources and formats. Python is popular for ETL workloads due to its many tools and frameworks (Russom, 2019). Python's ETL operations extract data from numerous sources, format it, and load it into a destination system, usually a database, for

storage and analysis.
Python's versatility and Pandas and NumPy's support make these tasks efficient. The extract phase retrieves data from SQL databases, cloud services, and flat files. The most important phase, transformation, includes cleaning, aggregating, and rearranging data, making Python's data manipulation skills indispensable (Rogel-Salazar, 2018).
Finally, the load step writes the changed data to a target data repository. Python is adaptable for this phase since it works with many database systems.

Orchid Tech uses Python for ETL processes because it needs a versatile and strong platform to manage varied data sources and complicated transformations. The target database system for these ETL processes, especially a relational database, affects data integration and analytics efficiency.

### 4.2 Benefits of Relational Databases

For decades, relational databases have been the standard for data storage and management. Relational databases' ACID (Atomicity, Consistency, Isolation, Durability) qualities provide consistent transactions and data integrity, making them ideal for Python ETL workloads, according to Date (2017). Orchid Tech needs reliable and consistent data across business functions.
Relational databases also handle organised business data well. SQL querying makes data retrieval easy (Codd, 2020). Orchid Tech's data analysis and reporting require fast, flexible data access.
Additionally, relational databases are well-supported and compatible with many tools and platforms, including Python. Silberschatz et al. (2021) stated this integration. simplifies data management by integrating Python's ETL procedures into the database.

### 4.3 Limitations and Challenges

Relational databases have drawbacks that can make Python ETL work difficult, especially in a fast-changing company like Orchid Tech. Scalability is an issue. Given Orchid Tech's expansion and growing data needs, relational databases may struggle with big volumes or high-velocity data influxes (Stonebraker, 2020).
Handling unstructured or semi-structured data, which is growing in the digital era, is another difficulty. Relational databases may hold unstructured data; however, they are not optimised for it, resulting in inefficiencies (Halevy et al., 2019).
Relational databases' inflexible schema can also be a negative. According to Cattell (2021), changing the database schema to support new data types or formats can be complicated and time-consuming, limiting Orchid Tech's agility in meeting changing data needs.

### 4.4 Other Options

Orchid Tech should investigate alternative data storage options due to relational database restrictions. NoSQL databases like MongoDB or Cassandra are alternatives. These databases

can handle enormous amounts of unstructured or semi-structured data and scale better than relational databases (Strauch et al., 2021).
Data warehousing solutions like Amazon Redshift or Google BigQuery are optimised for massive data and complicated queries, making them suited for analytical activities (Gupta, 2023). Scalability, performance, and flexibility are also benefits of cloud-based solutions.

Orchid Tech could also leverage polyglot persistence, integrating relational and NoSQL databases for varied needs (Sadalage and Fowler, 2023).

### 4.5 Final Evaluation

In conclusion, relational databases are resilient, reliable, and serve structured data well, but Orchid Tech must examine their scalability, unstructured data processing, and schema rigidity. As the organisation expands and its data needs become more complicated, relational databases may not be ideal for Python ETL jobs.

NoSQL databases or cloud-based data warehousing could scale and adapt to Orchid Tech's data needs. This does not mean abandoning relational databases. Orchid Tech may benefit most from a hybrid approach that balances relational and NoSQL databases to meet its broad and changing data management needs.

## 5. International Expansion: Big Data Issues

### 5.1 Big Data Challenge Overview

The international expansion of Orchid Tech Innovations Ltd. heralds' significant data management and interpretation. According to Mayer-Schönberger and Cukier (2023), big data is extraordinarily large data sets that can be computationally analysed to discover patterns, trends, and relationships, notably in human behaviour and interactions. As it enters new markets like Japan, the corporation must manage big data's three Vs: volume, velocity, and variety (Laney, 2021).
Orchid Tech's data will grow significantly. With new market expansion, customer interactions and transactional data will grow significantly. Effectively managing this massive amount of data is essential for company insights and choices.
Data velocity—the pace at which information is generated and processed—is another major issue. Fast processing is needed for real-time data like social media feeds and consumer interactions. Systems that can handle the constant data flow and deliver timely insights are needed.
Data variety is what Orchid Tech will experience. Unstructured data from social media and structured data from databases are included. Effective big data management requires managing this variety and making all data accessible and analysable.

## 5.2 Legal and Regulatory Considerations

Orchid Tech must navigate difficult foreign data protection laws as it expands abroad. According Krzysztofek (2018), the General Data Protection Regulation (GDPR) sets strict data privacy and security rules for companies handling EU residents' data. Orchid Tech must comply with GDPR by acquiring explicit consent for data collection and protecting data and privacy.
Japan's APPI oversees regulating data privacy. To comply with international standards like GDPR, APPI now protects personal data and regulates cross-border data transfers Marcén (2021). Orchid Tech must comply with APPI by properly handling Japanese clients' personal data, including consents and data security.

Global expansion requires compliance with each country's data protection rules. This entails understanding each market's laws and using data management procedures that match these various needs.

## 5.3 Scaling Operations: Technological Challenges

Orchid Tech faces huge data management issues as operations scale. The growing data volume requires scalable storage and processing. Distributed databases or cloud storage may be better for this scale than traditional database systems (Warren and Marz, 2015).
Orchid Tech needs real-time data processing technologies. Real-time analytics requires this since processing delays might cost opportunities or render insights obsolete. In this setting, in-memory databases and real-time data processing frameworks are vital (Kreps, 2014).

The variety of data makes integration and analysis difficult. Orchid Tech must analyse organised and unstructured data. This demands powerful data integration and analytics solutions that can handle varied data types (White, 2022).

Another major issue is international data transfer. Data transfer across borders requires compliance with data protection laws and data security. Data transfer must be secure, efficient, and compliant (Hon et al., 2022).

## 5.4 Big Data Challenges and Strategies

To overcome these issues, Orchid Tech needs extensive data governance regulations. Data governance ensures data quality, security, and legal compliance by creating data management policies and standards (Khatri and Brown, 2020). Orchid Tech should develop explicit data access, storage, processing, and sharing procedures to comply with international data protection legislation.

Technologically, Orchid Tech should use scalable cloud data storage and management systems. Cloud platforms scale and adapt to manage massive data sets (Warren and Marz, 2015). Apache Kafka and stream processing platforms can handle high-velocity data in real time (Kreps, 2014).

Orchid Tech needs advanced data integration technologies for varied data formats. Investment in big data analytics systems will allow the organisation to gain insights from many data sources (White, 2022).
Encryption and international data transfer standards are necessary for secure cross-border data transport. Orchid Tech should also consider data localization to comply with regional data protection legislation.

In a nutshell, Orchid Tech must use robust data governance, scalable technology, and international legal requirements to manage large amounts of data during international expansion.

## 6. General Recommendations

Based on Orchid Tech Innovations Ltd.'s IT infrastructure and big data management issues in global expansion, the following recommendations are made:

1. **Integrating ERP and CRM on the Cloud**: Orchid Tech should use a cloud-based ERP and CRM system. Addressing data silos and manual data integration, this integrated system will manage corporate processes, customer data, and operations across different locations, according to Mithas et al. (2023). A company looking to expand internationally needs scalability and flexibility, which a cloud-based solution provides.
2. **Hybrid Database Method**: Traditional relational databases struggle with massive data; hence, a hybrid approach is advised. Orchid Tech should use relational databases for structured data and NoSQL, or data warehousing, for unstructured or semi-structured data. While accepting various data formats, this strategy will balance data integrity and scalability (White, 2022).
3. **Data Processing in Real Time**: To manage big data velocity, use stream processing platforms or in-memory databases. These technologies, as noted by Kreps (2014), will allow Orchid Tech to process enormous data streams in real time for time-sensitive decisions.
4. **Strong Data Governance and Compliance**: Orchid Tech should create data governance policies to ensure data quality, security, and international data protection. This combines GDPR compliance for EU data and APPI compliance for Japanese customer data.
5. **Cybersecurity**: As data volume and cross-border data transit become more complex, better cybersecurity measures are essential. The list should contain encryption, safe data transport, and constant breach monitoring (Chen and Zhao, 2022).
6. **Training and change management**: Orchid Tech must alter significantly to implement these technology solutions. To enable a smooth transition to new systems and procedures, staff training and change management are essential.

## Conclusion

In conclusion, Orchid Tech Innovations Ltd.'s international and big data ventures offer enormous prospects and complicated obstacles. These issues involve handling an increasing amount of data, complying with foreign data protection laws, and adapting to global company technical needs.

Integrating a cloud-based ERP and CRM system, adopting a hybrid database approach, deploying real-time data processing technology, building robust data governance standards, and investing in cybersecurity are holistic solutions to these difficulties. These solutions improve Orchid Tech's data management, compliance, and operational efficiency.

Orchid Tech must adapt to the changing digital landscape during this transformation. Growing and staying competitive in the global market requires constant examination and change of these techniques. The successful adoption of these proposals will fix current problems and open new doors.

## REFERENCES


Cattell, R., 2021. Scalable SQL and NoSQL data stores. *Acm Sigmod Record*, *39*(4), pp.12-27.

Chen, D. and Zhao, H., 2022, March. Data security and privacy protection issues in cloud computing. In *2012 international conference on computer science and electronics engineering* (Vol. 1, pp. 647-651). IEEE.

Codd, E.F., 2020. A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), pp.377-387.

Date, C.J., 2017. *An introduction to database systems*. Pearson Education India.

Davenport, T., 2014. *Big data at work: dispelling the myths, uncovering the opportunities*. Harvard Business Review Press.

Dunie, R., Schulte, W.R., Cantara, M. and Kerremans, M., 2015. Magic Quadrant for intelligent business process management suites. *Gartner Inc*.

Fink, L. and Neumann, S., 2017. Gaining agility through IT personnel capabilities: The mediating role of IT infrastructure capabilities. *Journal of the Association for Information Systems*, *8*(8), p.25.

Gemawat, P., 2021. Distance still matters. *Harvard Business Review*, pp.137-147.

Gupta, N., 2023. Optimising data quality of a data warehouse using data purgation process. *International Journal of Data Mining, Modelling and Management*, *15*(1), pp.102-131.

Halevy, A., Norvig, P. and Pereira, F., 2019. The unreasonable effectiveness of data. *IEEE intelligent systems*, *24*(2), pp.8-12.

Hon, W.K., Millard, C. and Walden, I., 2022. Negotiating cloud contracts: Looking at clouds from both sides now. *Stan. Tech. L. Rev.*, *16*, p.79.

Khatri, V. and Brown, C.V., 2020. Designing data governance. *Communications of the ACM*, *53*(1), pp.148-152.

Kotler, P. and Keller, K., 2021. *Marketing Management (15th global edition)*. Pearson Education Limited.

Kreps, J., 2014. *I heart logs: Event data, stream processing, and data integration*. " O'Reilly Media, Inc.".

Krzysztofek, M., 2018. *GDPR: General Data Protection Regulation (EU) 2016/679: Post-reform Personal Data Protection in the European Union*. Kluwer Law International BV.

Laney, D., 2021. 3D data management: Controlling data volume, velocity and variety. *META group research note*, *6*(70), p.1.

Luftman, J.N., 2023. *Competing in the information age: Align in the sand*. Oxford University Press.

Marcén, A.G., 2021. The New Personal Data Protection in Japan: Is It Enough?. In *Media Technologies for Work and Play in East Asia* (pp. 101-120). Bristol University Press.

Mayer-Schönberger, V. and Cukier, K., 2023. *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt.

Mithas, S., Tafti, A. and Mitchell, W., 2023. How a firm's competitive environment and digital strategic posture influence digital business strategy. *MIS quarterly*, pp.511-536.

Peppard, J. and Ward, J., 2016. *The strategic management of information systems: Building a digital strategy*. John Wiley & Sons.

Rogel-Salazar, J., 2018. *Data science and analytics with Python*. CRC Press.

Rogers, M., 2020. *Managing Customer Relationships*. Wiley.

Russom, P., 2019. Next generation data warehouse platforms. *TDWI Best Practices Report: fourth quarter*.

Sadalage, P.J. and Fowler, M., 2023. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education.

Silberschatz, A., Korth, H.F. and Sudarshan, S., 2021. *Database system concepts*. McGraw-Hill.

Stonebraker, M., 2020. SQL databases v. NoSQL databases. *Communications of the ACM*, *53*(4), pp.10-11.

Strauch, C., Sites, U.L.S. and Kriha, W., 2021. NoSQL databases. *Lecture Notes, Stuttgart Media University*, *20*(24), p.79.

Themistocleous, M. and Chen, H., 2022. Investigating the integration of SMEs' information systems: an exploratory case study. *International Journal of Information Technology and Management*, *3*(2-4), pp.208-234.

Vassiliadis, P., 2019. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining (IJDWM)*, *5*(3), pp.1-27.

Warren, J. and Marz, N., 2015. *Big Data: Principles and best practices of scalable realtime data systems*. Simon and Schuster.

White, T., 2022. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.".

## Appendices

```
import csv

import json

import xml.etree.ElementTree as ET


# Function to read CSV file

def load_csv(file_path):

    with open(file_path, 'r', encoding='utf-8') as csvfile:

        return list(csv.DictReader(csvfile))


# Function to read JSON file

def load_json(file_path):

    with open(file_path, 'r', encoding='utf-8') as jsonfile:

        return json.loads(jsonfile.read())
```

```
# Function to read XML file
def load_xml(file_path):
    data = []
    tree = ET.parse(file_path)
    for elem in tree.getroot():
        data.append({child.tag: child.text for child in elem})
    return data

# Function to read TXT file
def load_txt(file_path):
    with open(file_path, 'r', encoding='utf-8') as txtfile:
        return [line.strip() for line in txtfile.readlines()]
def normalize_data(data, data_type):
    normalized = []
    for record in data:
        if data_type == 'csv' or data_type == 'xml':
            normalized.append(process_record(record))
        elif data_type == 'json':
            normalized.append(process_json_record(record))
        elif data_type == 'txt':
            normalized.append({'text': record})
    return normalized

def process_record(record):
    return {key: clean_value(value) for key, value in record.items()}

def process_json_record(record):
    return {key: clean_value(value) for key, value in record.items()}
```

```
def clean_value(value):
    if isinstance(value, str) and value.isdigit():
        return int(value)
    return value.strip() if isinstance(value, str) else value
def merge_datasets(*args):
    merged = {}
    for dataset in args:
        for item in dataset:
            key = item.get('id') or item.get('customer_id') or item.get('text', '')
            if key not in merged:
                merged[key] = item
            else:
                merged[key].update(item)
    return list(merged.values())
from pony.orm import Database, Required, Optional, PrimaryKey, db_session

db = Database()

class Customer(db.Entity):
    id = PrimaryKey(int, auto=True)
    firstName = Optional(str)
    lastName = Optional(str)
    age = Optional(int)
    sex = Optional(str)
    vehicleMake = Optional(str)
    vehicleModel = Optional(str)
    vehicleYear = Optional(int)
    vehicleType = Optional(str)
    iban = Optional(str)
```

```
    creditCardNumber = Optional(str)
    creditCardSecurityCode = Optional(str)
    creditCardStartDate = Optional(str)  # Assuming string format; adjust if using datetime
    creditCardEndDate = Optional(str)  # Assuming string format; adjust if using datetime
    addressMain = Optional(str)
    addressCity = Optional(str)
    addressPostcode = Optional(str)
    retired = Optional(str)
    dependants = Optional(int)
    marital_status = Optional(str)
    salary = Optional(float)
    pension = Optional(float)
    company = Optional(str)
    commute_distance = Optional(float)
    text = Optional(str)  # Assuming this is for the TXT file data

db.bind(provider='sqlite', filename=':memory:', create_db=True)
db.generate_mapping(create_tables=True)

@db_session
def insert_into_db(data):
    for record in data:
        Customer(**record)  # Insert each record into the database
def normalize_data(data, data_type):
    normalized = []
    for record in data:
        normalized_record = {}
        if data_type == 'csv':
            normalized_record = {
```

```
            'firstName': record.get('First Name', '').strip(),
            'lastName': record.get('Second Name', '').strip(),
            # Add mappings for other fields
        }
    elif data_type == 'json':
        normalized_record = {
            'firstName': record.get('firstName', '').strip(),
            'lastName': record.get('lastName', '').strip(),
            # Add mappings for other fields
        }
    elif data_type == 'xml':
        normalized_record = {
            'firstName': record.get('firstName', '').strip(),
            'lastName': record.get('lastName', '').strip(),
            # Add mappings for other fields
        }
    elif data_type == 'txt':
        # Handle TXT data as required
        normalized_record = {'text': record}

    normalized.append(normalized_record)
  return normalized
# Example usage
csv_data_example = [{'First Name': 'Michael', 'Second Name': 'Fadele'}]
normalized_csv_data = normalize_data(csv_data_example, 'csv')
print(normalized_csv_data)
import json
def read_json(file_path):
  with open(file_path, 'r', encoding='utf-8') as file:
```

```
        return json.load(file)
import xml.etree.ElementTree as ET
def read_xml(file_path):
    tree = ET.parse(file_path)
    root = tree.getroot()
    data = [{child.tag: child.text for child in user} for user in root]
    return data
def read_txt(file_path):
    data = []
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            # Assuming a specific format for the TXT file, such as JSON-like or key-value pairs
            # This needs to be adjusted based on the actual format of your TXT data.
            entry = eval(line.strip())
            data.append(entry)
    return data
def normalize_record(record):
    normalized = {}
    # Example of normalization: renaming fields, handling missing data, etc.
    normalized['customer_id'] = record.get('id') or record.get('customer_id')
    normalized['first_name'] = record.get('firstName', '').strip()
    normalized['last_name'] = record.get('lastName', '').strip()
    normalized['email'] = record.get('email', None)  # Assuming email might not be present in
all records
    # Add more fields as necessary based on your data structure


    return normalized
def normalize_data(data):
    return [normalize_record(record) for record in data]
```

```
def merge_data(datasets):
    """Merge multiple datasets into one.

    Args:
        datasets (list of list of dict): List of datasets to merge.

    Returns:
        list of dict: Merged dataset.
    """
    merged = {}
    for data in datasets:
        for record in data:
            customer_id = record['customer_id']
            if customer_id in merged:
                # Update existing record with any new information
                merged[customer_id].update({k: v for k, v in record.items() if v})
            else:
                # Add new record
                merged[customer_id] = record
    return list(merged.values())
from pony.orm import Database, Required, Optional, PrimaryKey, db_session

db = Database()

class Customer(db.Entity):
    customer_id = PrimaryKey(str)
    first_name = Required(str)
    last_name = Required(str)
    email = Optional(str)
```

```
    # Add other fields as necessary

db.bind(provider='sqlite', filename=':memory:', create_db=True)
db.generate_mapping(create_tables=True)
```