Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

БАЗА ДАННЫХ ERP СИСТЕМЫ

БГУИР КР 1-40 01 01 115 ПЗ

Студент: гр. 251001 Лашкин В.Н.

Руководитель:

асс. Фадеева Е.Е.

Минск 2025

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

––––––––––––––––––––––––

(подпись)

––––––––––––––––– 2025 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Лашкину Владиславу Николаевичу

1. Тема работы “База данных ERP системы”

2. Срок сдачи студентом законченной работы 16.05.2025 г.–––

3. Исходные данные к работе

- Описание основных бизнес-процессов ERP-системы

- Требования к структуре базы данных

- СУБД MySQL

4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;

2. Анализ требований к программному средству и разработка функциональных требований;

3. Инфологическая модель предметной области;

4. Подробное описание бизнес-логики;

5. Тестирование, проверка работоспособности и анализ полученных результатов;

Список используемой литературы

Заключение

Приложение А

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. "База данных ERP системы", А1, модель данных, чертеж

2. "База данных ERP системы", А1, схема данных, чертеж

6. Консультант по курсовой работе

Фадеева Е.Е.

7. Дата выдачи задания 21.01.2025

8. Календарный график работы над курсовой работой на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1 к 01.02.2025 – 15 % готовности работы;

разделы 2, 3 к 01.03.2025 – 30 % готовности работы;

разделы 4, 5 к 01.04.2025 – 60 % готовности работы;

раздел 6 к 01.05.2025 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 10.05.2025 – 100 % готовности работы.

Защита курсовой работы с 12.05.2025 по 16.05.2025 г.

РУКОВОДИТЕЛЬ–––––– Е.Е.Фадеева

(подпись)

Задание принял к исполнению –––\_\_\_\_ ––

(дата и подпись студента)

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 4](#_Toc198147976)

[ВВЕДЕНИЕ 6](#_Toc198147977)

[1. АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ 7](#_Toc198147978)

[1.1. Общие сведения 7](#_Toc198147979)

[1.2. Примеры решения аналогичных задач, анализ достоинств и недостатков известных решений 8](#_Toc198147980)

[1.3. Постановка задачи 11](#_Toc198147981)

[1.3.1. Назначение разработки 11](#_Toc198147982)

[1.3.2. Состав выполняемых функций 11](#_Toc198147983)

[1.3.3. Входные данные 12](#_Toc198147984)

[1.3.4. Выходные данные 12](#_Toc198147985)

[1.3.5. Требования к составу параметрам технических и программных средств 13](#_Toc198147986)

[1.3.6. Требования к информационной и программной совместимости 13](#_Toc198147987)

[2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 15](#_Toc198147988)

[2.1. Обоснование выбора программных средств 15](#_Toc198147989)

[2.2. Учитываемые атрибуты, функции и бизнес-процессы 16](#_Toc198147990)

[2.3. Описание функциональных требований 17](#_Toc198147991)

[2.4. Ключевые особенности будущей БД 18](#_Toc198147992)

[2.5. Пользователи системы и их роли 18](#_Toc198147993)

[2.6. Параметры поиска и другой обработки информации 19](#_Toc198147994)

[2.7. Бизнес-процессы, связанные с реализацией системы 19](#_Toc198147995)

[2.8. Обоснование выбора метода взаимодействия клиента и сервера 20](#_Toc198147996)

[3. ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ 21](#_Toc198147997)

[3.1. Сущности и связи 21](#_Toc198147998)

[3.2. Особенности нормализации 27](#_Toc198147999)

[4. ПОДРОБНОЕ ОПИСАНИЕ БИЗНЕС-ЛОГИКИ 28](#_Toc198148000)

[4.1. Основные представления 28](#_Toc198148001)

[4.2. Основные триггеры 29](#_Toc198148002)

[4.3. Основные хранимые процедуры и функции 31](#_Toc198148003)

[5. ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 36](#_Toc198148004)

[5.1. Тестирование хранимых процедур 36](#_Toc198148005)

[5.2. Тестирование триггеров 38](#_Toc198148006)

[5.3. Тестирование хранимых функций 38](#_Toc198148007)

[5.4. Тестирование представлений 39](#_Toc198148008)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 41](#_Toc198148009)

[ЗАКЛЮЧЕНИЕ 42](#_Toc198148010)

[ПРИЛОЖЕНИЕ А 43](#_Toc198148011)

# ВВЕДЕНИЕ

В современном бизнесе автоматизация процессов играет ключевую роль в повышении эффективности работы компаний. ERP-системы (Enterprise Resource Planning) позволяют объединить различные бизнес-процессы, включая управление персоналом, складом, финансами, закупками и продажами, в единую информационную среду.

Основой любой ERP-системы является база данных, которая обеспечивает хранение, обработку и доступ к информации. От правильной структуры базы данных зависит производительность системы, ее масштабируемость и удобство использования. Разработка базы данных ERP-системы требует тщательного анализа бизнес-процессов, нормализации данных и создания взаимосвязанных таблиц, обеспечивающих целостность информации.

Цель данной курсовой работы — разработка структуры базы данных для ERP-системы, охватывающей ключевые модули предприятия. В ходе работы будут рассмотрены требования к системе, создана схема базы данных и реализованы SQL-скрипты для работы с ней.

Разработка базы данных ERP-системы является актуальной задачей, поскольку качественная организация данных способствует эффективному управлению ресурсами предприятия, снижению издержек и повышению конкурентоспособности.

# АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

## Общие сведения

ERP (Enterprise Resource Planning) – это комплексное программное обеспечение, предназначенное для автоматизации и интеграции ключевых бизнес-процессов предприятия. ERP-системы охватывают различные аспекты деятельности организации, включая управление финансами, складскими запасами, производством, персоналом, продажами и закупками.

Основной целью ERP-систем является создание единой информационной среды, обеспечивающей согласованность и доступность данных для всех подразделений компании. Это позволяет повысить прозрачность бизнес-процессов, улучшить управляемость ресурсами и сократить затраты на их учет и контроль.

Ключевые компоненты ERP-системы:

1. Управление финансами – учет доходов и расходов, расчет налогов, контроль платежей и формирование финансовой отчетности.
2. Управление персоналом – учет сотрудников, ведение кадрового делопроизводства, расчет заработной платы.
3. Складской учет – контроль запасов, управление движением товаров между складами.
4. Закупки и поставки – ведение заказов поставщикам, контроль выполнения договоров.
5. Продажи и клиенты (CRM) – учет заказов, взаимодействие с клиентами, управление ценообразованием.
6. Производственный учет – планирование и контроль производства, управление ресурсами.

ERP-системы применяются в различных отраслях: производстве, торговле, логистике, строительстве, здравоохранении и других сферах. Их использование позволяет автоматизировать рутинные операции, минимизировать ошибки, улучшить принятие управленческих решений и повысить конкурентоспособность предприятия.

Основой любой ERP-системы является база данных, которая обеспечивает надежное хранение и обработку большого объема информации. Корректное проектирование базы данных играет ключевую роль в эффективности работы ERP-системы, влияя на быстродействие, масштабируемость и безопасность данных.

## Примеры решения аналогичных задач, анализ достоинств и недостатков известных решений

**Odoo**

Odoo (ранее известная как Tiny ERP и OpenERP) — ERP- и CRM-система, разработанная бельгийской компанией Odoo S.A. Система распространяется по лицензии LGPL и предназначена для предприятий малого и среднего бизнеса.

A screenshot of a computer

AI-generated content may be incorrect.

Рисунок 1.1 – Интерфейс Odoo

Основные функции:

* Бухгалтерский учет
* Управление взаимоотношениями с клиентами (CRM)
* Управление персоналом
* Производство
* Продажи и закупки
* Складской учет
* Управление проектами
* Электронная коммерция

Плюсы:

* Открытый исходный код, позволяющий адаптировать систему под специфические потребности компании.​
* Широкий набор модулей и приложений, обеспечивающих гибкость и масштабируемость.​
* Активное сообщество разработчиков и пользователей, предоставляющее поддержку и регулярные обновления.​

Минусы:

* Некоторые модули могут быть менее функциональными по сравнению с аналогами в других ERP-системах.​
* Требуется время и ресурсы на настройку и адаптацию системы под конкретные бизнес-процессы.

**Oracle E-Business Suite**

Oracle E-Business Suite (OEBS) — интегрированный комплекс прикладного программного обеспечения от компании Oracle, включающий функциональные блоки ERP, CRM и PLM. Предназначен для автоматизации основных направлений деятельности предприятий, включая финансы, производство, управление персоналом и логистику.

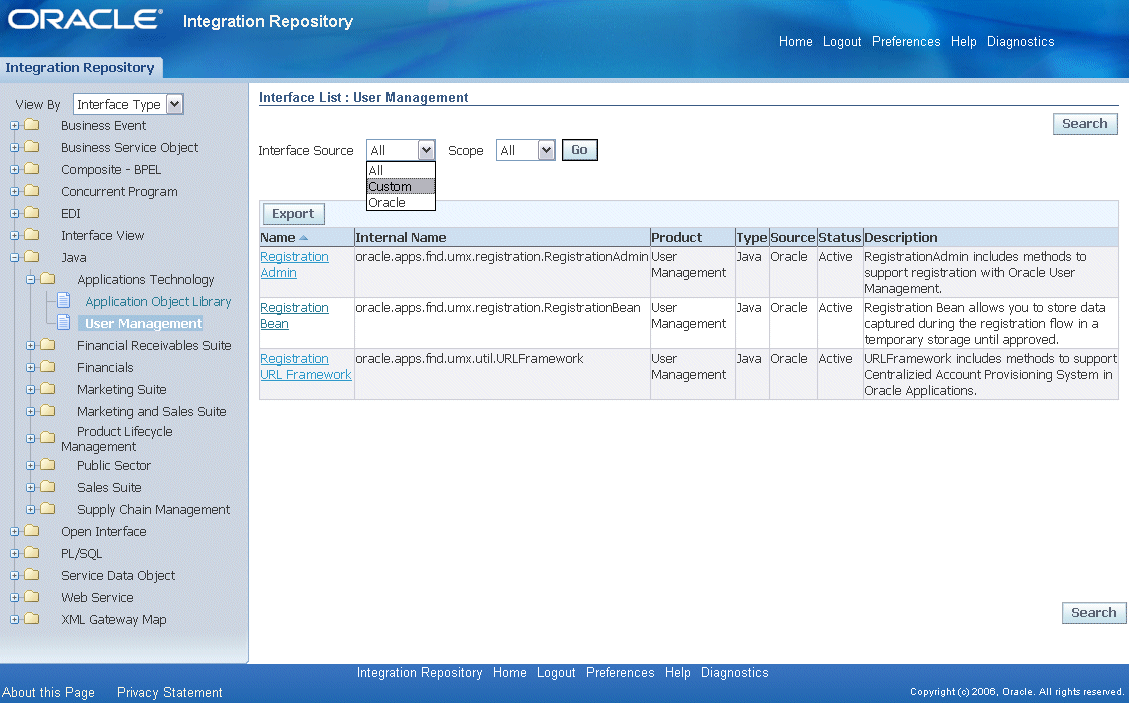


Рисунок 1.2 – Интерфейс Oracle E-Business Suite

Основные функции:

* Финансовый менеджмент​
* Управление цепочками поставок​
* Управление взаимоотношениями с клиентами​
* Управление персоналом​
* Производственное планирование​
* Управление проектами​

Плюсы:

* Глубокая интеграция модулей, обеспечивающая целостность данных и процессов.​
* Высокая производительность и масштабируемость, подходящая для крупных предприятий.​
* Широкий спектр функциональных возможностей, охватывающих различные аспекты бизнеса.​

Минусы:

* Высокая стоимость лицензий и внедрения, что может быть неподъемным для малого и среднего бизнеса.​

**Microsoft Dynamics 365**

Microsoft Dynamics AX (ныне известная как Dynamics 365) — ERP-система от корпорации Microsoft, предназначенная для среднего и крупного бизнеса. Система предоставляет функции финансового менеджмента, бизнес-анализа и управления производственными процессами.

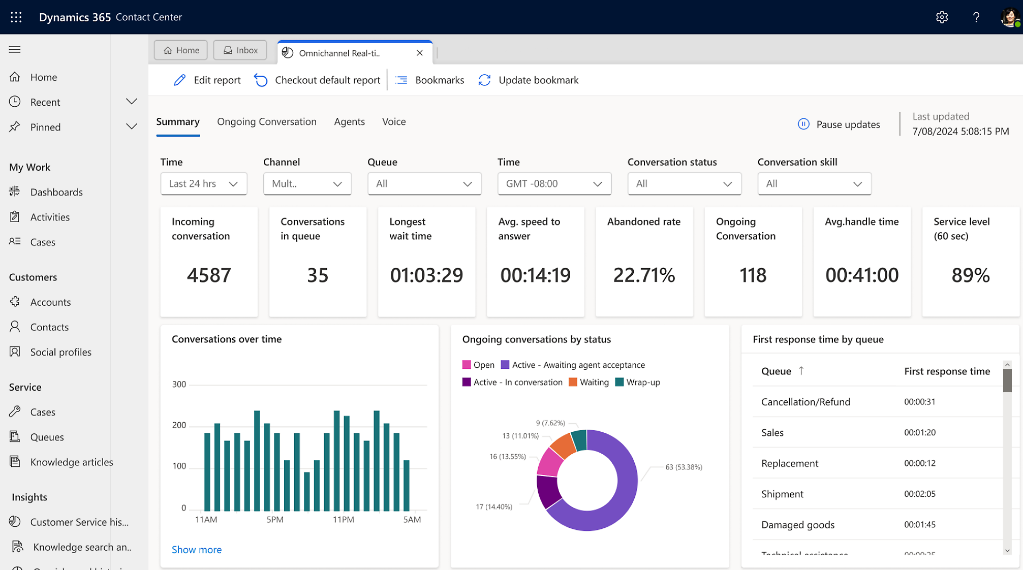


Рисунок 1.3 – Интерфейс Microsoft Dynamics 365

Основные функции:

* Финансовый менеджмент​
* Управление производством​
* Управление цепочками поставок​
* Управление проектами​
* Управление персоналом​
* Продажи и маркетинг​

Плюсы:

* Интеграция с другими продуктами Microsoft, такими как Office 365, обеспечивающая удобство использования.​
* Гибкость настройки и возможность адаптации под различные отраслевые требования.​
* Поддержка облачных решений, позволяющая снизить затраты на инфраструктуру.​

Минусы:

* Высокая стоимость лицензий и внедрения.​
* Сложность миграции с предыдущих версий и интеграции с системами сторонних производителей.

## Постановка задачи

### Назначение разработки

Разрабатываемая база данных предназначена для использования в ERP-системе предприятия, ориентированной на автоматизированный учет, хранение и обработку информации, связанной с ключевыми бизнес-процессами организации. Система обеспечивает поддержку процессов управления заказами, клиентами, товарами, закупками, платежами, персоналом и складскими запасами, а также предоставляет сотрудникам, администраторам и аналитикам доступ к актуальной информации о деятельности компании.

### Состав выполняемых функций

База данных должна предоставлять ERP-системе возможность выполнения следующих функций:

1. Организация работы администратора системы:
   1. Управление пользователями и ролями сотрудников;
   2. Контроль и аудит бизнес-процессов через систему логирования;
   3. Настройка справочников (единицы измерения, статусы, типы оплат и т.п.);
   4. Управление подразделениями и структурами предприятия;
   5. Назначение прав доступа к различным функциям системы;
   6. Просмотр аналитической информации и отчётности.
2. Организация работы менеджера по продажам:
   1. Создание и управление заказами клиентов;
   2. Добавление товаров в заказы и формирование счетов;
   3. Отслеживание статусов заказов;
   4. Проведение оплат и контроль задолженностей клиентов;
   5. Ведение клиентской базы.
3. Организация работы склада:
   1. Учёт остатков товаров на складах;
   2. Проведение инвентаризаций и перемещений между складами;
   3. Приёмка товаров по закупочным заказам;
   4. Формирование и обновление информации по наличию продукции.
4. Организация работы отдела закупок:
   1. Создание и управление закупочными заказами;
   2. Контроль поставок от поставщиков;
   3. Учёт стоимости закупок и прогнозирование потребностей;
   4. Ведение базы поставщиков.
5. Регистрация и авторизация сотрудников:
   1. Регистрация сотрудников различных ролей (администраторов, менеджеров, кладовщиков);
   2. Авторизация и контроль активности пользователей в системе.
6. Поддержка аналитики и отчетности:
   1. Отображение финансовых и складских отчётов;
   2. Формирование сводной информации по заказам, клиентам и поставщикам;
   3. Просмотр динамики продаж и остатков;
   4. Поддержка бизнес-решений через агрегированные представления данных.

### Входные данные

Входными данными для разрабатываемой базы данных ERP-системы являются: сотрудники, должности, отделы, роли пользователей, клиенты, заказы клиентов, товары, категории товаров, единицы измерения, склады, остатки товаров на складах, поставщики, закупочные заказы, позиции закупок, статусы заказов, статусы поставок, счета-фактуры, платежи, способы оплаты, налоговые ставки, скидки, акты приёмки, перемещения товаров между складами, производственные операции, партии товаров, финансовые операции, типы операций, лог активности пользователей, доступы пользователей, бизнес-события, уведомления, отчёты, расписания поставок, оценки поставщиков, задачи персонала, начисления заработной платы, графики работы сотрудников, рабочие календари, шаблоны документов, архивные записи заказов, история изменений статусов, системные параметры, журналы ошибок, пользовательские действия, внутренние заявки, история перемещений товаров, назначенные менеджеры, контактные лица клиентов, контактные лица поставщиков, отзывы клиентов, возвраты товаров, причины возврата, и связанные с ними документы.

### Выходные данные

Выходными данными для разрабатываемой ERP-системы являются: сведения о заказах клиентов, включая статус, дату и общую сумму, информация о текущем балансе клиентов по всем их счетам, данные об остатках товаров на складах, агрегированные сведения о ежедневных продажах, контактные данные и полные имена сотрудников, информация о статусах закупочных заказов и их общей стоимости, финансовая сводка по выставленным счетам и произведённым платежам, список заказов, по которым требуется выставление счёта, история изменений статусов заказов, записи о поступивших платежах от клиентов, информация о новых пользователях, зарегистрированных в системе, сведения об изменениях остатков на складе после обновлений или поставок, журнал действий пользователей в системе, сформированные заказы, их состав, счета и соответствующие оплаты, сведения о перемещении товаров между складами.

### Требования к составу параметрам технических и программных средств

1. Аппаратные требования:
   1. Процессор: не менее 2 ядер с тактовой частотой от 2 ГГц;
   2. Оперативная память: не менее 4 ГБ;
   3. Накопитель: SSD-диск с минимум 20 ГБ свободного пространства;
   4. Сетевое подключение: стабильное интернет-соединение при работе в удалённой/облачной среде.
2. Программные требования:
   1. Операционная система: Windows 10/11, Ubuntu 22.04+, macOS 12+;
   2. Система управления базами данных: MySQL 8.0 или совместимая (например, MariaDB 10.6);
   3. Средства работы с БД: MySQL Workbench, DBeaver, HeidiSQL или phpMyAdmin.

### 1.3.6. Требования к информационной и программной совместимости

1. Информационная совместимость:
   1. Использование кодировки UTF-8 (utf8mb4) для обеспечения корректной работы с многоязычным контентом;
   2. Форматы хранения данных, совместимые с другими корпоративными системами (например, CSV, JSON, XML);
   3. Возможность масштабирования структуры базы данных под изменяющиеся бизнес-процессы.
2. Программная совместимость:
   1. Поддержка работы на основных платформах: Windows, Linux, macOS;
   2. Возможность размещения в облачных инфраструктурах: AWS (Amazon RDS), Google Cloud SQL, Azure Database for MySQL, DigitalOcean Managed Databases;
   3. Интеграция с внешними приложениями и сервисами через API и поддержка взаимодействия с языками программирования, такими как Java, C#, PHP, Python, JavaScript;
   4. Совместимость с системами контроля версий (Git) и CI/CD-пайплайнами при развёртывании в корпоративной среде.

# АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

## Обоснование выбора программных средств

При разработке базы данных для ERP-системы были выбраны следующие программные средства: MySQL в качестве системы управления базами данных (СУБД), MySQL Workbench для проектирования и администрирования базы данных, а также dbdiagram.io для визуального моделирования структуры данных.

1. Преимущества MySQL:
   1. Популярность и зрелость — MySQL является одной из самых широко используемых реляционных СУБД, что гарантирует стабильность и наличие большого количества обучающих материалов;
   2. Открытость и бесплатность — используется MySQL Community Edition, не требующая лицензирования;
   3. Производительность и масштабируемость — обеспечивает высокую производительность и возможность масштабирования при увеличении объёмов данных;
   4. Поддержка транзакций и ограничений целостности — позволяет надёжно управлять критичными бизнес-операциями в рамках ERP-системы.
2. Преимущества MySQL Workbench:
   1. Интуитивно понятный интерфейс для работы с таблицами, представлениями, индексами и пользователями;
   2. Удобное написание и отладка SQL-запросов благодаря встроенному редактору с подсветкой синтаксиса и автодополнением;
   3. Визуальное проектирование схемы БД — возможность создавать и изменять структуру данных с помощью диаграмм;
   4. Средства диагностики и администрирования — включают в себя мониторинг производительности и анализ нагрузок.
3. Преимущества dbdiagram.io:
   1. Простота создания ER-диаграмм в текстовом формате с мгновенной визуализацией;
   2. Быстрое редактирование схем без установки дополнительного ПО;
   3. Генерация SQL-кода по диаграммам для удобного импорта в СУБД;
   4. Возможность совместной работы — поддержка экспорта, импорта и публикации моделей.

## Учитываемые атрибуты, функции и бизнес-процессы

В разрабатываемой базе данных ERP-системы отражены ключевые атрибуты и сущности, необходимые для автоматизации внутренних процессов организации. Система охватывает следующие основные аспекты бизнес-деятельности:

1. Учитываемые атрибуты и объекты:
   1. Пользователи системы — идентификаторы, контактные данные, роли и права доступа;
   2. Клиенты и поставщики — наименования, реквизиты, история заказов и оплат;
   3. Заказы и счета — статусы, даты оформления, суммы, связанные товары и услуги;
   4. Товары и складские запасы — названия, артикула, остатки, цены, единицы измерения;
   5. Сотрудники — ФИО, должности, контакты, активность в системе;
   6. Платежи и финансовые транзакции — суммы, даты, типы операций;
   7. Производственные и закупочные документы — спецификации, стоимости, статусы выполнения.
2. Учитываемые функции и действия:
   1. Управление заказами клиентов и поставками от поставщиков;
   2. Отслеживание остатков на складе и движения товаров;
   3. Расчёт общей стоимости заказов и остатков в режиме реального времени;
   4. Регистрация и логирование изменений статусов и ключевых действий;
   5. Учёт финансовых операций, включая выставление счетов и проведение платежей;
   6. Поддержка рабочих ролей: администратор, менеджер, кладовщик и бухгалтер.
3. Бизнес-процессы, моделируемые системой:
   1. Приём и обработка заказов клиентов;
   2. Закупка и приёмка товаров на склад;
   3. Выдача товаров и списание остатков;
   4. Управление статусами заказов и поставок;
   5. Ведение финансового учёта: оплата счетов, поступления и расходы;
   6. Ведение пользовательской и технической документации: логирование, аудит изменений;
   7. Подготовка аналитических данных — отчёты о продажах, остатках, задолженностях.

## Описание функциональных требований

Разрабатываемая база данных ERP-системы предназначена для хранения и управления ключевой информацией, связанной с пользователями, ролями, документами, уведомлениями и действиями сотрудников. Функциональные требования определяют, какие задачи должна выполнять система, и каким образом пользователи будут взаимодействовать с её модулями.

Основные функциональные требования:

1. Управление пользователями и ролями:
   1. Система должна хранить информацию о пользователях (ФИО, логин, email, статус и т.д.).
   2. Каждый пользователь должен иметь привязанную роль, определяющую его уровень доступа (например, администратор, сотрудник и т.д.).
   3. Пользователь может быть активным или деактивированным, что влияет на возможность входа в систему.
2. Управление документами:
   1. Система должна позволять хранить документы с основными атрибутами (название, описание, дата создания, тип документа и т.д.).
   2. Каждый документ должен иметь автора, связанного с конкретным пользователем.
   3. Документы должны классифицироваться по типу и быть связаны с определёнными тегами для облегчения поиска и фильтрации.
   4. Для документа необходимо отслеживать текущий статус (например, «в работе», «утверждён», «отклонён»).
3. Назначение ответственных лиц:
   1. Для каждого документа должен быть назначен один или несколько ответственных пользователей.
   2. Требуется фиксировать дату назначения и возможность замены ответственного.
4. Тегирование и классификация:
   1. Каждый документ может быть связан с одним или несколькими тегами, которые позволяют группировать и фильтровать документы по тематикам.
5. Уведомления
   1. Система должна создавать уведомления для пользователей о событиях (например, назначение ответственного, изменение статуса документа и т.д.).
   2. Уведомления должны содержать тип события, дату создания и ID связанного документа или пользователя.
6. Аудит действий:
   1. Необходимо вести журнал действий пользователей (например, вход в систему, создание или редактирование документа, изменение статуса и т.д.).
   2. Для каждой записи аудита фиксируются ID пользователя, тип действия, дата и связанный объект (например, документ или уведомление).

## Ключевые особенности будущей БД

База данных для разрабатываемой ERP-системы должна решать две ключевые задачи.

Во-первых, она должна обеспечивать централизованный учёт всех внутренних бизнес-процессов компании. В базе фиксируются заказы, поставки, запасы на складах, расчёты с клиентами и поставщиками, данные о сотрудниках и их действиях, а также финансовые показатели. Система должна быть гибкой, чтобы адаптироваться под особенности структуры предприятия и поддерживать работу различных подразделений: отдела закупок, отдела продаж, склада, бухгалтерии и управления персоналом.

Во-вторых, база данных должна учитывать интеграцию с внешними интерфейсами и модулями. ERP-система предполагает наличие административной панели, клиентской зоны и интерфейса для поставщиков. Пользователи системы (например, сотрудники компании или контрагенты) могут работать с данными через онлайн-интерфейс, при этом информация о действиях автоматически синхронизируется и логируется в базе данных.

## Пользователи системы и их роли

Система предполагает участие следующих пользователей с их основными задачами:

1. Администратор:
   1. Имеет полный доступ ко всем сущностям и настройкам базы данных;
   2. Управляет пользователями, ролями и правами доступа;
   3. Настраивает справочники системы (категории товаров, статусы заказов, типы платежей и др.);
   4. Контролирует работу всех модулей и осуществляет аудит изменений.
2. Менеджер по продажам:
   1. Ведёт клиентскую базу, добавляет и редактирует карточки клиентов;
   2. Оформляет и изменяет заказы клиентов;
   3. Контролирует статусы заказов и взаимодействует с бухгалтерией по вопросам оплаты;
   4. Формирует коммерческие предложения и рассчитывает итоговую стоимость заказа.
3. Кладовщик:
   1. Учёт поступления товаров на склады и оформление приёмочных документов;
   2. Отслеживание остатков и проведение инвентаризаций;
   3. Перемещение товаров между складами;
   4. Подготовка отгрузок по заказам и оформление отгрузочных документов.
4. Бухгалтер:
   1. Формирование и отправка клиентам счетов-фактур;
   2. Учёт поступивших платежей и сверка по задолженностям;
   3. Ведение налогового учёта и расчёт налоговых обязательств;
   4. Подготовка финансовой отчётности и аналитических сводок.
5. Гость (неавторизованный пользователь):
   1. Просмотр каталога товаров, прайс-листов, публичной информации о компании;
   2. Ознакомление с условиями доставки и оплаты;
   3. Подача предварительных запросов на получение коммерческого предложения.

## Параметры поиска и другой обработки информации

Основной операцией в ERP-системе является поиск и фильтрация данных, поэтому необходимо обеспечить высокую производительность запросов по ключевым атрибутам.

1. Наиболее часто поиск будет осуществляться по следующим полям:
   1. Статусы сущностей (фильтрация заказов, счетов, поставок, заявок по статусам);
   2. Характеристики товаров (категория, артикул, наименование, диапазон цен, поставщик);
   3. Информация о контрагентах (название клиента или поставщика, ИНН, контактное лицо);
   4. Даты (дата заказа, дата поставки, дата выставления счета, срок оплаты, даты перемещений на складе);
   5. Сотрудники (ФИО, должность, подразделение) — для быстрого поиска исполнителей задач и ответственных лиц.

## Бизнес-процессы, связанные с реализацией системы

При внедрении ERP-системы учитываются следующие ключевые процессы:

1. Приём заказа от клиента — регистрация заказа, проверка остатков и кредитных лимитов, подтверждение менеджером.
2. Сборка и отгрузка товаров — планирование задачи кладовщику, оформление перемещений на складе, генерация отгрузочных документов.
3. Управление закупками — оформление закупочных заказов поставщикам, приёмка и учёт поступивших товаров.
4. Инвентаризация и перемещения — регулярная сверка фактических и учётных остатков, внутренние перемещения между складами.
5. Финансовые операции — выставление счетов, регистрация оплат, контроль дебиторской и кредиторской задолженности.
6. Отчётность и аналитика — формирование ежедневных, еженедельных и ежемесячных отчётов по продажам, закупкам, остаткам и денежным потокам.
7. Логирование и аудит — автоматическая регистрация ключевых действий (изменение статусов, операции с остатками, авторизация пользователей).

## Обоснование выбора метода взаимодействия клиента и сервера

Для обмена данными между клиентским приложением и сервером баз данных выбран протокол HTTP/HTTPS с архитектурой REST:

1. Стандартность и совместимость. HTTP-запросы (GET, POST, PUT, DELETE) широко поддерживаются на всех платформах и упрощают интеграцию с внешними сервисами.
2. Безопасность. HTTPS с SSL/TLS обеспечивает шифрование трафика и защиту конфиденциальных данных при передаче.
3. Простота и масштабируемость. REST-подход позволяет независимо масштабировать фронтенд и бэкенд, а также легко добавлять новые ресурсы.
4. В качестве формата обмена данными используется JSON:
5. Читаемость. JSON-документы легко интерпретируются как человеком, так и машиной.
6. Универсальность. Поддерживается большинством языков программирования и сред разработки.
7. Лёгкость парсинга. Простая структура ускоряет передачу и обработку данных на сервере и клиенте.

# ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

## Сущности и связи

Инфологическая модель определяет основные объекты предметной области и их взаимосвязи, которые лягут в основу структуры базы данных. В данном разделе представлены ключевые сущности системы, их атрибуты и связи, обеспечивающие целостность и корректность данных. Эти сущности отражают основные бизнес-процессы ERP-системы. Ниже приведена таблица с описанием основных сущностей.

Таблица 3.1 – Описание основных сущностей и их связей

|  |  |  |  |
| --- | --- | --- | --- |
| Отношение | Описание | Основные атрибуты | Краткое описание связей с другими отношениями |
| users | Хранит информацию о пользователях системы. | Id, username, password\_hash, email, created\_at, updated\_at | Связаны с roles через user\_roles; участвуют в logs, sessions, notifications, audit\_trail |
| roles | Определяет роли, которые могут быть присвоены пользователям. | Id, name, description, created\_at | Связаны с users через user\_roles; с permissions через role\_permissions |
| user\_roles | Связующая таблица между пользователями и ролями. | User\_id, role\_id | Связывает users и roles (многие ко многим) |
| permissions | Список всех разрешений в системе. | Id, name, description | Связаны с roles через role\_permissions |
| role\_permissions | Связующая таблица между ролями и разрешениями. | role\_id, permission\_id | Связывает roles и permissions (многие ко многим) |
| organizations | Организации, к которым могут принадлежать отделы. | Id, name, address, created\_at | Связаны с departments |

Таблица 3.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| departments | Отделы внутри организаций. | Id, organization\_id, name, created\_at | Связаны с organizations и employees через employee\_departments |
| employees | Сотрудники компании. | Id, first\_name, last\_name, email, phone, hire\_date, created\_at | Связаны с departments через employee\_departments; с timesheets, expenses, project\_tasks, maintenance\_requests |
| employee\_departments | Связь между сотрудниками и отделами. | Employee\_id, department\_id | Связывает employees и departments (многие ко многим) |
| customers | Клиенты компании. | Id, company\_name, contact\_name, contact\_email, created\_at | Связаны с orders |
| suppliers | Поставщики товаров или услуг. | Id, company\_name, contact\_name, contact\_email, created\_at | Связаны с products и purchase\_orders |
| categories | Категории продуктов. | Id, name, description | Связаны с products и product\_categories |
| products | Продукты, продаваемые или закупаемые компанией. | Id, supplier\_id, category\_id, sku, name, description, unit\_price, created\_at | Связаны с suppliers, categories, order\_items, invoice\_items, shipment\_items, inventory, price\_list\_items, purchase\_order\_items |

Таблица 3.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| product\_categories | Дополнительная связь продуктов и категорий (многие ко многим). | Product\_id, category\_id | Связывает products и categories (многие ко многим) |
| orders | Заказы, сделанные клиентами. | Id, customer\_id, order\_date, status, created\_at | Связаны с customers, order\_items, invoices, shipments |
| order\_items | Позиции, входящие в заказы. | Order\_id, product\_id, quantity, unit\_price | Связаны с orders и products; используются в invoice\_items и shipment\_items |
| invoices | Счета, выставленные на основе заказов. | Id, order\_id, invoice\_date, due\_date, total\_amount, created\_at | Связаны с orders, invoice\_items, payments |
| invoice\_items | Позиции в счетах-фактурах. | Invoice\_id, order\_item\_order\_id, order\_item\_product\_id, quantity, unit\_price | Связаны с invoices и order\_items |
| payment\_methods | Методы оплаты (наличные, карта и т.д.). | Id, method\_name, details | Используются в payments |
| payments | Оплаты по счетам. | Id, invoice\_id, amount, payment\_method\_id, created\_at | Связаны с invoices и payment\_methods |
| shipments | Отгрузки заказов клиентам. | Id, order\_id, shipment\_date, carrier, tracking\_number, created\_at | Связаны с orders и shipment\_items |

Таблица 3.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| shipment\_items | Товары, включенные в отгрузку. | Shipment\_id, order\_item\_order\_id, order\_item\_product\_id, quantity | Связаны с shipments и order\_items |
| warehouses | Склады, где хранятся товары. | Id, name, location | Связаны с inventory |
| inventory | Остатки товаров на складах. | Product\_id, warehouse\_id, quantity\_on\_hand | Связывает products и warehouses |
| purchase\_orders | Заказы на закупку товаров у поставщиков. | Id, supplier\_id, order\_date, status, created\_at | Связаны с suppliers и purchase\_order\_items |
| purchase\_order\_items | Позиции в заказах на закупку. | Purchase\_order\_id, product\_id, quantity, unit\_cost | Связаны с purchase\_orders и products |
| price\_lists | Прайс-листы с ценами на товары. | Id, name, effective\_date | Связаны с price\_list\_items |
| price\_list\_items | Цены на товары в рамках прайс-листов. | Price\_list\_id, product\_id, price | Связывает price\_lists и products |
| currencies | Поддерживаемые валюты. | Code\_name\_symbol | Связаны с exchange\_rates |
| exchange\_rates | Курсы обмена между валютами. | From\_currency, to\_currency, rate, date | Связывает две валюты (from\_currency, to\_currency) |
| taxation\_rules | Налоговые правила, применяемые к операциям. | Id, name, rate, applicable\_from | Неявно связаны с операциями (например, orders, invoices) |

Таблица 3.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| projects | Проекты, управляемые в компании. | Id, name, start\_date, end\_date, status | Связаны с project\_tasks |
| project\_tasks | Задачи, связанные с проектами. | Id, project\_id, name, assignee\_id, due\_date, status | Связаны с projects, employees (assignee\_id), timesheets |
| timesheets | Учет рабочего времени сотрудников по задачам. | Employee\_id, project\_task\_id, date, hours | Связаны с employees и project\_tasks |
| expense\_categories | Категории расходов. | Id, name | Связаны с expenses |
| expenses | Фактические расходы сотрудников. | Id, employee\_id, expense\_category\_id, amoung, incurred\_on | Связаны с employees и expense\_categories |
| assets | Активы, находящиеся в собственности компании. | Id, name, purchase\_date, value | Связаны с asset\_movements, maintenance\_requests |
| asset\_movements | Перемещения активов между локациями. | Asset\_id, from\_location, to\_location, moved\_on | Связаны с assets |
| maintenance\_requests | Запросы на обслуживание активов. | Id, asset\_id, requested\_by, requested\_on, status | Связаны с assets, employees (requested\_by), maintenance\_logs |
| maintenance\_logs | Журнал работ по обслуживанию активов. | Id, request\_id, logged\_on, notes | Связаны с maintenance\_requests |

Таблица 3.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| contacts | Контактная информация для разных сущностей. | Id, entity\_type, entity\_id, contact\_type, contact\_value | Связываются с разными сущностями через entity\_type и entity\_id |
| addresses | Адреса, связанные с сущностями. | Id, entity\_type, entity\_id, address\_line1, address\_line2, city, state, postal\_code, country | Связываются с разными сущностями через entity\_type и entity\_id |
| logs | Журнал действий пользователей в системе. | Id, user\_id, action, entity, entity\_id, created\_at | Связаны с users |
| notifications | Уведомления, создаваемые системой. | Id, title, body, created\_at | Связаны с notification\_recipients |
| notification\_recipients | Получатели уведомлений. | Notification\_id, user\_id, read | Связывают notifications и users |
| sessions | Сессии пользователей | Id, user\_id, created\_at, expires\_at | Связаны с users |
| configurations | Таблица с конфигурациями для приложения | Id, config\_key, config\_value | Не связана явно с другими таблицами |
| audit\_trail | Таблица для аудита | Id, table\_name, record\_id, action, changed\_by, changed\_at | Связана с любой таблицей через table\_name и record\_id; указывает, кто изменил (changed\_by — user) |

Таблица 3.1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| system\_settings | Таблица с системными настройками | Key, value | Не связана явно с другими таблицами |
| message\_templates | Шаблоны для сообщений | Id, name, subject | Не связана явно с другими таблицами |

## Особенности нормализации

В процессе проектирования базы данных ERP-системы были приняты следующие решения по обеспечению нормализации и целостности данных.

1. Атомарность атрибутов
   1. Имя и фамилия сотрудников, клиентов и поставщиков представлены отдельными полями first\_name и last\_name для упрощения поиска, сортировки и локализации по фамилии.
   2. Адресные данные вынесены в отдельную сущность addresses и разбиты на логические компоненты (address\_line1, city, country и др.), чтобы избежать дублирования и облегчить хранение многострочных адресов.
   3. Контактные данные (contact\_type, contact\_value) вынесены в таблицу contacts, что позволяет хранить для одной сущности произвольное число способов связи (телефон, email, социальные сети) без нарушения принципа атомарности.
2. Разделение сущностей
   1. Заказы (orders) и позиции заказов (order\_items) хранятся в разных таблицах, так как один заказ может содержать множество товарных строк.
   2. Счета (invoices) и позиции счетов (invoice\_items) также выделены в отдельные таблицы для точного отражения деталей расчетов и поддержки нескольких позиций на одном счете.
   3. Клиенты (customers) и поставщики (suppliers) хранятся в отдельных таблицах, несмотря на сходную структуру, чтобы разграничить процессы продажи и закупки.
   4. Товары (products) и категории товаров (categories), а также связь многие-ко-многим (product\_categories) разделены для гибкой классификации и возможности назначения нескольких категорий одному товару.
   5. Уведомления (notifications) и получатели (notification\_recipients) вынесены в отдельные таблицы.

# ПОДРОБНОЕ ОПИСАНИЕ БИЗНЕС-ЛОГИКИ

## Основные представления

1. v\_order\_overview

Показывает общую информацию по заказам: клиент, дата, статус и сумма заказа.

CREATE VIEW `v\_order\_overview` AS

SELECT o.id AS order\_id, c.company\_name, o.order\_date, o.status,

       fn\_calculate\_order\_total(o.id) AS total\_amount

FROM orders o

JOIN customers c ON o.customer\_id = c.id;

1. v\_customer\_balances

Отображает текущие балансы клиентов, рассчитанные с помощью функции.

CREATE VIEW `v\_customer\_balances` AS

SELECT c.id AS customer\_id, c.company\_name,

       fn\_get\_customer\_balance(c.id) AS balance

FROM customers c;

1. v\_inventory\_levels

Показывает остатки товаров на складах по каждому продукту.

CREATE VIEW `v\_inventory\_levels` AS

SELECT p.id AS product\_id, p.name, fn\_get\_stock\_level(p.id) AS quantity\_on\_hand

FROM products p;

1. v\_sales\_summary

Сводка продаж по дням с суммарной выручкой за каждый день.

CREATE VIEW `v\_sales\_summary` AS

SELECT DATE(o.order\_date) AS sale\_date,

       SUM(oi.quantity \* oi.unit\_price) AS daily\_sales

FROM orders o

JOIN order\_items oi ON o.id = oi.order\_id

GROUP BY DATE(o.order\_date);

1. v\_employee\_directory

Краткий справочник сотрудников с ФИО, email и телефоном.

CREATE VIEW `v\_employee\_directory` AS

SELECT id AS employee\_id, fn\_get\_employee\_fullname(id) AS fullname, email, phone

FROM employees;

1. v\_purchase\_order\_status

Информация по закупкам: поставщик, дата, статус и сумма заказа.

CREATE VIEW `v\_purchase\_order\_status` AS

SELECT po.id, s.company\_name AS supplier, po.order\_date, po.status,

       SUM(poi.quantity \* poi.unit\_cost) AS total\_cost

FROM purchase\_orders po

JOIN suppliers s ON po.supplier\_id = s.id

JOIN purchase\_order\_items poi ON po.id = poi.purchase\_order\_id

GROUP BY po.id;

## Основные триггеры

1. trg\_users\_after\_insert

Сохраняет запись в журнал аудита после добавления нового пользователя, включая все данные вставленной строки.

DELIMITER $$

CREATE TRIGGER `trg\_users\_after\_insert`

AFTER INSERT ON `users`

FOR EACH ROW

BEGIN

  INSERT INTO `audit\_trail`(`table\_name`,`record\_id`,`action`,`changed\_by`)

  VALUES('users', NEW.id, 'INSERT', NEW.id);

END$$

DELIMITER ;

1. trg\_inventory\_after\_update

Записывает в аудит старые и новые значения при обновлении записей об остатках товаров.

DELIMITER $$

CREATE TRIGGER `trg\_inventory\_after\_update`

AFTER UPDATE ON `inventory`

FOR EACH ROW

BEGIN

  INSERT INTO `audit\_trail`(`table\_name`,`record\_id`,`action`,`changed\_by`)

  VALUES('inventory', OLD.product\_id, 'UPDATE', NULL);

END$$

DELIMITER ;

1. trg\_orders\_status\_change

Логирует изменение статуса заказа перед обновлением, если статус действительно изменился.

DELIMITER $$

CREATE TRIGGER `trg\_orders\_status\_change`

BEFORE UPDATE ON `orders`

FOR EACH ROW

BEGIN

  IF OLD.status <> NEW.status THEN

    INSERT INTO `logs`(`user\_id`,`action`,`entity`,`entity\_id`)

    VALUES(NULL, CONCAT('Order status changed from ', OLD.status, ' to ', NEW.status), 'orders', NEW.id);

  END IF;

END$$

DELIMITER ;

1. trg\_payments\_after\_insert

Создаёт лог-запись после внесения платежа, включая сумму и номер счёта.

DELIMITER $$

CREATE TRIGGER `trg\_payments\_after\_insert`

AFTER INSERT ON `payments`

FOR EACH ROW

BEGIN

  INSERT INTO `logs`(`user\_id`,`action`,`entity`,`entity\_id`)

  VALUES(NULL, CONCAT('Payment of ', NEW.amount, ' recorded for invoice ', NEW.invoice\_id), 'payments', NEW.id);

END$$

DELIMITER ;

1. trg\_purchase\_order\_after\_insert

Автоматически обновляет складские остатки при поступлении товаров по закупке — добавляет или увеличивает количество.

DELIMITER $$

CREATE TRIGGER `trg\_purchase\_order\_after\_insert`

AFTER INSERT ON `purchase\_order\_items`

FOR EACH ROW

BEGIN

  INSERT INTO `inventory`(`product\_id`,`warehouse\_id`,`quantity\_on\_hand`)

  VALUES(NEW.product\_id, 1, NEW.quantity)

  ON DUPLICATE KEY UPDATE quantity\_on\_hand = quantity\_on\_hand + NEW.quantity;

END$$

DELIMITER ;

## Основные хранимые процедуры и функции

1. sp\_create\_order

Создаёт новый заказ с заданной датой и клиентом, присваивает ему статус "New" и возвращает идентификатор созданного заказа.

DELIMITER $$

CREATE PROCEDURE `sp\_create\_order`(

  IN p\_customer\_id INT,

  IN p\_order\_date DATE,

  OUT p\_order\_id INT

)

BEGIN

  INSERT INTO `orders`(`customer\_id`,`order\_date`,`status`)

  VALUES(p\_customer\_id, p\_order\_date, 'New');

  SET p\_order\_id = LAST\_INSERT\_ID();

END$$

DELIMITER ;

1. sp\_add\_product\_to\_order

Добавляет товар в указанный заказ, автоматически подставляя его текущую цену из справочника продуктов.

DELIMITER $$

CREATE PROCEDURE `sp\_add\_product\_to\_order`(

  IN p\_order\_id INT,

  IN p\_product\_id INT,

  IN p\_quantity INT

)

BEGIN

  DECLARE unit\_price DECIMAL(10,2);

  SELECT `unit\_price` INTO unit\_price FROM `products` WHERE id = p\_product\_id;

  INSERT INTO `order\_items`(`order\_id`,`product\_id`,`quantity`,`unit\_price`)

  VALUES(p\_order\_id, p\_product\_id, p\_quantity, unit\_price);

END$$

DELIMITER ;

1. sp\_record\_payment

Регистрирует оплату по счёту, включая дату, сумму и способ оплаты.

DELIMITER $$

CREATE PROCEDURE `sp\_record\_payment`(

  IN p\_invoice\_id INT,

  IN p\_amount DECIMAL(12,2),

  IN p\_method\_id INT

)

BEGIN

INSERT INTO `payments`(`invoice\_id`,`payment\_date`,`amount`,`payment\_method\_id`)

  VALUES(p\_invoice\_id, CURDATE(), p\_amount, p\_method\_id);

END$$

DELIMITER ;

1. sp\_transfer\_stock

Выполняет перемещение товара с одного склада на другой с соответствующим обновлением остатков.

DELIMITER $$

CREATE PROCEDURE `sp\_transfer\_stock`(

  IN p\_product\_id INT,

  IN p\_from\_wh INT,

  IN p\_to\_wh INT,

  IN p\_quantity INT

)

BEGIN

  UPDATE `inventory` SET quantity\_on\_hand = quantity\_on\_hand - p\_quantity

    WHERE product\_id = p\_product\_id AND warehouse\_id = p\_from\_wh;

  INSERT INTO `inventory`(`product\_id`,`warehouse\_id`,`quantity\_on\_hand`)

    VALUES(p\_product\_id, p\_to\_wh, p\_quantity)

    ON DUPLICATE KEY UPDATE quantity\_on\_hand = quantity\_on\_hand + p\_quantity;

END$$

DELIMITER ;

1. sp\_create\_invoice

Создаёт счёт на основе заказа, рассчитывая итоговую сумму и устанавливая дату оплаты и срок.

DELIMITER $$

CREATE PROCEDURE `sp\_create\_invoice`(

  IN p\_order\_id INT,

  OUT p\_invoice\_id INT

)

BEGIN

  DECLARE total DECIMAL(12,2) DEFAULT 0;

  SELECT SUM(quantity \* unit\_price) INTO total FROM `order\_items` WHERE order\_id = p\_order\_id;

  INSERT INTO `invoices`(`order\_id`,`invoice\_date`,`due\_date`,`total\_amount`)

  VALUES(p\_order\_id, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 30 DAY), total);

  SET p\_invoice\_id = LAST\_INSERT\_ID();

END$$

DELIMITER ;

1. sp\_assign\_employee\_to\_department

Назначает сотрудника в отдел, избегая повторного добавления в ту же связь.

DELIMITER $$

CREATE PROCEDURE `sp\_assign\_employee\_to\_department`(

  IN p\_employee\_id INT,

  IN p\_department\_id INT

)

BEGIN

  INSERT IGNORE INTO `employee\_departments`(`employee\_id`,`department\_id`)

  VALUES(p\_employee\_id, p\_department\_id);

END$$

DELIMITER ;

1. fn\_calculate\_order\_total

Вычисляет итоговую сумму заказа по идентификатору, суммируя стоимость всех позиций (количество × цена за единицу):

DELIMITER $$

CREATE FUNCTION `fn\_calculate\_order\_total`(p\_order\_id INT)

RETURNS DECIMAL(12,2)

DETERMINISTIC

BEGIN

DECLARE total DECIMAL(12,2);

SELECT SUM(quantity \* unit\_price) INTO total FROM `order\_items` WHERE order\_id = p\_order\_id;

RETURN IFNULL(total, 0);

END$$

DELIMITER ;

1. fn\_calculate\_tax

Рассчитывает сумму налога, умножая сумму на ставку налога и округляя результат до 2 знаков:

DELIMITER $$

CREATE FUNCTION `fn\_calculate\_tax`(p\_amount DECIMAL(12,2), p\_tax\_rate DECIMAL(5,4))

RETURNS DECIMAL(12,2)

DETERMINISTIC

RETURN ROUND(p\_amount \* p\_tax\_rate, 2);$$

DELIMITER ;

1. fn\_format\_currency

Форматирует денежное значение, добавляя символ валюты и два знака после запятой.

DELIMITER $$

CREATE FUNCTION `fn\_format\_currency`(p\_amount DECIMAL(12,2), p\_symbol VARCHAR(10))

RETURNS VARCHAR(50)

DETERMINISTIC

RETURN CONCAT(p\_symbol, FORMAT(p\_amount, 2));$$

DELIMITER ;

1. fn\_get\_customer\_balance

Определяет текущую задолженность клиента, вычитая сумму оплат из общей суммы счетов.

DELIMITER $$

CREATE FUNCTION `fn\_get\_customer\_balance`(p\_customer\_id INT)

RETURNS DECIMAL(12,2)

DETERMINISTIC

BEGIN

DECLARE bal DECIMAL(12,2);

SELECT IFNULL(SUM(i.total\_amount) - SUM(IFNULL(p.amount, 0)), 0)

INTO bal

FROM `invoices` i

LEFT JOIN `payments` p ON i.id = p.invoice\_id

WHERE i.order\_id IN (SELECT id FROM `orders` WHERE customer\_id = p\_customer\_id);

RETURN bal;

END$$

DELIMITER ;

1. fn\_get\_stock\_level

Возвращает текущее количество товара на складе по его идентификатору.

DELIMITER $$

CREATE FUNCTION `fn\_get\_stock\_level`(p\_product\_id INT)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE qty INT;

SELECT SUM(quantity\_on\_hand) INTO qty FROM `inventory` WHERE product\_id = p\_product\_id;

RETURN IFNULL(qty, 0);

END$$

DELIMITER ;

1. fn\_get\_employee\_fullname

Формирует полное имя сотрудника, объединяя имя и фамилию.

DELIMITER $$

CREATE FUNCTION `fn\_get\_employee\_fullname`(p\_emp\_id INT)

RETURNS VARCHAR(101)

DETERMINISTIC

BEGIN

DECLARE fname VARCHAR(50);

DECLARE lname VARCHAR(50);

SELECT first\_name, last\_name INTO fname, lname FROM `employees` WHERE id = p\_emp\_id;

RETURN CONCAT(fname, ' ', lname);

END$$

DELIMITER ;

1. fn\_days\_between

Возвращает количество дней между двумя датами.

DELIMITER $$

CREATE FUNCTION `fn\_days\_between`(p\_start DATE, p\_end DATE)

RETURNS INT

DETERMINISTIC

RETURN DATEDIFF(p\_end, p\_start);$$

DELIMITER ;

# ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

## Тестирование хранимых процедур

Таблица 5.1 – Тестирование хранимых процедур

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тестируемая процедура | Спецификация тестирования | Номер теста | Ожидаемый результат | Полученный результат |
| sp\_create\_order | Создание заказа с валидным customer\_id и датой | 1 | Новая запись в orders, возвращён корректный order\_id | Тест пройден |
| Попытка создания заказа с несуществующим customer\_id | 2 | Ошибка внешнего ключа или пустой результат | Тест пройден |
| sp\_add\_product\_to\_order | Добавление товара в существующий заказ | 3 | Новая запись в order\_items с верными unit\_price | Тест пройден |
| Добавление товара с несуществующим product\_id | 4 | Ошибка внешнего ключа | Тест пройден |
| sp\_record\_payment | Регистрация платежа по валидному invoice\_id | 5 | Новая запись в payments с сегодняшней датой | Тест пройден |
| Регистрация платежа по несуществующему invoice\_id | 6 | Ошибка внешнего ключа | Тест пройден |

Таблица 5.1 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sp\_transfer\_stock | Перемещение доступного товара между складами | 7 | Остаток на складе-отправителе уменьшен, на складе-получателе увеличен | Тест пройден |
| Перемещение количества, превышающего остаток на складе | 8 | Ошибка проверки или отрицательный остаток | Тест пройден |
| sp\_create\_invoice | Создание счёта для существующего заказа | 9 | Новая запись в invoices с корректным total\_amount | Тест пройден |
| Создание счёта для заказа без позиций | 10 | Счёт с total\_amount = 0 | Тест пройден |
| sp\_assign\_employee\_to\_department | Назначение сотрудника в отдел впервые | 11 | Новая запись в employee\_departments | Тест пройден |
| Повторное назначение того же сотрудника в тот же отдел | 12 | Безошибочное выполнение, дублирующая запись не создана | Тест пройден |

## Тестирование триггеров

Таблица 5.2 – Тестирование триггеров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тестируемая процедура | Спецификация тестирования | Номер теста | Ожидаемый результат | Полученный результат |
| trg\_users\_after\_insert | Вставка нового пользователя | 13 | Запись в audit\_trail с table\_name = 'users' и action = 'INSERT' | Тест пройден |
| trg\_inventory\_after\_update | Обновление остатка товара | 14 | Запись в audit\_trail с action = 'UPDATE', JSON diff | Тест пройден |
| trg\_orders\_status\_change | Изменение статуса заказа | 15 | Запись в logs с текстом «Order status changed from … to …» | Тест пройден |
| trg\_payments\_after\_insert | Регистрация нового платежа | 16 | Запись в logs с деталями платежа | Тест пройден |

## Тестирование хранимых функций

Таблица 5.3 – Тестирование хранимых функций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тестируемая процедура | Спецификация тестирования | Номер теста | Ожидаемый результат | Полученный результат |
| fn\_calculate\_order\_total | Сумма по существующему заказу | 17 | Корректная сумма | Тест пройден |
| Сумма по заказу без позиций | 18 | 0 | Тест пройден |
| fn\_calculate\_tax | Расчёт налога для заданной суммы и ставки | 19 | Округлённый результат | Тест пройден |

Таблица 5.3 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fn\_format\_currency | Форматирование суммы с символом | 20 | Строка «<символ>X,XXX.XX» | Тест пройден |
| fn\_get\_customer\_balance | Баланс клиента с существующими счетами и платежами | 21 | Разница счетов и платежей | Тест пройден |
| Баланс клиента без операций | 22 | 0 | Тест пройден |
| fn\_get\_stock\_level | Остаток по существующему товару | 23 | Сумма остатков | Тест пройден |
| Остаток по товару, которого нет в inventory | 24 | 0 | Тест пройден |
| fn\_get\_employee\_fullname | Получение ФИО по существующему сотруднику | 25 | «Имя Фамилия» | Тест пройден |
| fn\_days\_between | Разница дат (более 0) | 26 | Корректное число дней | Тест пройден |

## Тестирование представлений

Таблица 5.4 – Тестирование представлений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тестируемая процедура | Спецификация тестирования | Номер теста | Ожидаемый результат | Полученный результат |
| v\_order\_overview | Существующие заказы | 27 | Строки с order\_id, company\_name, total\_amount | Тест пройден |
| Заказы отсутствуют | 28 | Пустой результат | Тест пройден |
| v\_customer\_balances | Клиенты с задолженностью | 29 | Строки с customer\_id, balance | Тест пройден |
| Клиенты без операций | 30 | balance = 0 | Тест пройден |

Таблица 5.4 – Продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| v\_inventory\_levels | Товары с остатками | 31 | product\_id, quantity\_on\_hand | Тест пройден |
| Товары отсутствуют в inventory | 32 | quantity\_on\_hand = 0 | Тест пройден |
| v\_sales\_summary | Продажи за определённую дату | 33 | sale\_date, daily\_sales | Тест пройден |
| v\_employee\_directory | Сотрудники существуют | 34 | employee\_id, fullname, email, phone | Тест пройден |
| v\_purchase\_order\_status | Закупочные заказы с позициями | 35 | supplier, total\_cost | Тест пройден |

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

[1] Куликов, С. Реляционные базы данных в примерах. – Минск, 2021. – 424 с.

[2] Куликов, С. Работа с MySQL, MS SQL Server и Oracle в примерах. – Минск, 2021. – 600 с.

[3] ГОСТ 7.0.100-2018 "Библиографическая запись. Библиографическое описание. Общие требования и правила составления" [Электронный ресурс]. Режим доступа: https://docs.cntd.ru/document/1200161674

[4] Репозиторий БГУИР [Электронный ресурс]. Режим доступа: https://libeldoc.bsuir.by/

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была спроектирована и реализована полноценная структура базы данных для ERP-системы, охватывающая ключевые бизнес-процессы организации: управление заказами, клиентами, продуктами, поставками, платежами, персоналом и складскими остатками. Система включает 50 взаимосвязанных таблиц, обеспечивающих целостность и логическую связность данных. Кроме того, были разработаны и внедрены триггеры, хранимые процедуры, функции и представления, позволяющие автоматизировать действия, повысить безопасность операций и упростить доступ к агрегированной информации.

При проектировании особое внимание уделялось соответствию высоким стандартам качества: все таблицы создаются с учётом зависимостей, используются механизмы аудита, логирования и актуализации данных. Проект легко масштабируем, поддерживает развитие функционала и может быть основой для полноценной ERP-системы.

Таким образом, поставленные цели и задачи были успешно реализованы, а созданная база данных может эффективно использоваться как в учебных, так и в практических целях.

# ПРИЛОЖЕНИЕ А

CREATE DATABASE IF NOT EXISTS `erp`

USE `erp`;

CREATE TABLE IF NOT EXISTS `addresses` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `entity\_type` varchar(50) NOT NULL,

  `entity\_id` int(11) NOT NULL,

  `address\_line1` varchar(255) NOT NULL,

  `address\_line2` varchar(255) DEFAULT NULL,

  `city` varchar(100) DEFAULT NULL,

  `state` varchar(100) DEFAULT NULL,

  `postal\_code` varchar(20) DEFAULT NULL,

  `country` varchar(100) DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `addresses` (`id`, `entity\_type`, `entity\_id`, `address\_line1`, `address\_line2`, `city`, `state`, `postal\_code`, `country`) VALUES

  (1, 'users', 1, '123 Maple St', NULL, 'CityA', 'StateA', '12345', 'USA'),

  (2, 'customers', 1, '456 Oak St', 'Suite 100', 'CityB', 'StateB', '23456', 'USA');

CREATE TABLE IF NOT EXISTS `assets` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `purchase\_date` date DEFAULT NULL,

  `value` decimal(12,2) DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `assets` (`id`, `name`, `purchase\_date`, `value`) VALUES

  (1, 'Laptop', '2024-01-01', 1500.00),

  (2, 'Printer', '2023-05-15', 300.00);

CREATE TABLE IF NOT EXISTS `asset\_movements` (

  `asset\_id` int(11) NOT NULL,

  `from\_location` varchar(100) DEFAULT NULL,

  `to\_location` varchar(100) DEFAULT NULL,

  `moved\_on` datetime NOT NULL,

  PRIMARY KEY (`asset\_id`,`moved\_on`),

  CONSTRAINT `asset\_movements\_ibfk\_1` FOREIGN KEY (`asset\_id`) REFERENCES `assets` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `asset\_movements` (`asset\_id`, `from\_location`, `to\_location`, `moved\_on`) VALUES

  (1, 'Office', 'Home', '2025-01-10 09:00:00'),

  (2, 'Warehouse', 'Office', '2025-01-11 10:00:00');

CREATE TABLE IF NOT EXISTS `audit\_trail` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `table\_name` varchar(100) NOT NULL,

  `record\_id` int(11) NOT NULL,

  `action` varchar(50) NOT NULL,

  `changed\_by` int(11) DEFAULT NULL,

  `changed\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `changed\_by` (`changed\_by`),

  CONSTRAINT `audit\_trail\_ibfk\_1` FOREIGN KEY (`changed\_by`) REFERENCES `users` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `audit\_trail` (`id`, `table\_name`, `record\_id`, `action`, `changed\_by`, `changed\_at`) VALUES

  (1, 'users', 1, 'INSERT', 1, '2025-05-14 05:57:58'),

  (2, 'users', 2, 'INSERT', 2, '2025-05-14 05:57:58'),

  (3, 'users', 3, 'INSERT', 3, '2025-05-14 05:57:58'),

  (4, 'users', 4, 'INSERT', 4, '2025-05-14 05:57:58'),

  (5, 'inventory', 1, 'UPDATE', NULL, '2025-05-14 05:57:58'),

  (6, 'inventory', 2, 'UPDATE', NULL, '2025-05-14 05:57:58'),

  (7, 'inventory', 1, 'UPDATE', NULL, '2025-05-14 05:57:58'),

  (8, 'inventory', 2, 'UPDATE', NULL, '2025-05-14 05:57:58'),

  (9, 'users', 1, 'UPDATE', 1, '2025-05-14 05:58:19'),

  (10, 'orders', 1, 'CREATE', 2, '2025-05-14 05:58:19'),

  (11, 'inventory', 4, 'UPDATE', NULL, '2025-05-14 07:14:10'),

  (12, 'users', 5, 'INSERT', 5, '2025-05-14 07:20:02');

CREATE TABLE IF NOT EXISTS `categories` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `description` text DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `categories` (`id`, `name`, `description`) VALUES

  (1, 'Category A', 'Description A'),

  (2, 'Category B', 'Description B'),

  (3, 'Category C', 'Description C'),

  (4, 'Category D', 'Description D');

CREATE TABLE IF NOT EXISTS `configurations` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `config\_key` varchar(100) NOT NULL,

  `config\_value` text DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `config\_key` (`config\_key`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `configurations` (`id`, `config\_key`, `config\_value`) VALUES

  (1, 'site\_name', 'MySite'),

  (2, 'support\_email', 'support@example.com');

CREATE TABLE IF NOT EXISTS `contacts` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `entity\_type` varchar(50) NOT NULL,

  `entity\_id` int(11) NOT NULL,

  `contact\_type` varchar(50) DEFAULT NULL,

  `contact\_value` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `contacts` (`id`, `entity\_type`, `entity\_id`, `contact\_type`, `contact\_value`) VALUES

  (1, 'users', 1, 'phone', '555-0100'),

  (2, 'customers', 1, 'email', 'alice.cust@example.com'),

  (3, 'suppliers', 1, 'email', 'sam.sup@example.com');

CREATE TABLE IF NOT EXISTS `currencies` (

  `code` char(3) NOT NULL,

  `name` varchar(50) NOT NULL,

  `symbol` varchar(10) DEFAULT NULL,

  PRIMARY KEY (`code`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `currencies` (`code`, `name`, `symbol`) VALUES

  ('EUR', 'Euro', '€'),

  ('GBP', 'British Pound', '£'),

  ('USD', 'US Dollar', '$');

CREATE TABLE IF NOT EXISTS `customers` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `company\_name` varchar(100) NOT NULL,

  `contact\_name` varchar(100) DEFAULT NULL,

  `contact\_email` varchar(100) DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `customers` (`id`, `company\_name`, `contact\_name`, `contact\_email`, `created\_at`) VALUES

  (1, 'Customer A', 'Alice Contact', 'alice.cust@example.com', '2025-01-05 06:00:00'),

  (2, 'Customer B', 'Bob Contact', 'bob.cust@example.com', '2025-01-06 06:00:00'),

  (3, 'Customer C', 'Carol Contact', 'carol.cust@example.com', '2025-01-07 06:00:00'),

  (4, 'Customer D', 'Dave Contact', 'dave.cust@example.com', '2025-01-08 06:00:00');

CREATE TABLE IF NOT EXISTS `departments` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `organization\_id` int(11) NOT NULL,

  `name` varchar(100) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `organization\_id` (`organization\_id`),

  CONSTRAINT `departments\_ibfk\_1` FOREIGN KEY (`organization\_id`) REFERENCES `organizations` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `departments` (`id`, `organization\_id`, `name`, `created\_at`) VALUES

  (1, 1, 'HR', '2025-01-01 07:00:00'),

  (2, 1, 'IT', '2025-01-01 07:30:00'),

  (3, 2, 'Sales', '2025-01-02 07:00:00'),

  (4, 3, 'Support', '2025-01-03 07:00:00');

CREATE TABLE IF NOT EXISTS `employees` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `first\_name` varchar(50) NOT NULL,

  `last\_name` varchar(50) NOT NULL,

  `email` varchar(100) NOT NULL,

  `phone` varchar(20) DEFAULT NULL,

  `hire\_date` date DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  UNIQUE KEY `email` (`email`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `employees` (`id`, `first\_name`, `last\_name`, `email`, `phone`, `hire\_date`, `created\_at`) VALUES

  (1, 'John', 'Smith', 'john.smith@example.com', '555-0100', '2024-06-01', '2025-01-01 08:00:00'),

  (2, 'Jane', 'Doe', 'jane.doe@example.com', '555-0101', '2024-07-15', '2025-01-02 08:00:00'),

  (3, 'Jim', 'Beam', 'jim.beam@example.com', '555-0102', '2024-08-20', '2025-01-03 08:00:00'),

  (4, 'Jill', 'Stark', 'jill.stark@example.com', '555-0103', '2024-09-10', '2025-01-04 08:00:00');

CREATE TABLE IF NOT EXISTS `employee\_departments` (

  `employee\_id` int(11) NOT NULL,

  `department\_id` int(11) NOT NULL,

  PRIMARY KEY (`employee\_id`,`department\_id`),

  KEY `department\_id` (`department\_id`),

  CONSTRAINT `employee\_departments\_ibfk\_1` FOREIGN KEY (`employee\_id`) REFERENCES `employees` (`id`) ON DELETE CASCADE,

  CONSTRAINT `employee\_departments\_ibfk\_2` FOREIGN KEY (`department\_id`) REFERENCES `departments` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `employee\_departments` (`employee\_id`, `department\_id`) VALUES

  (1, 1),

  (1, 2),

  (2, 2),

  (3, 3),

  (4, 4);

CREATE TABLE IF NOT EXISTS `exchange\_rates` (

  `from\_currency` char(3) NOT NULL,

  `to\_currency` char(3) NOT NULL,

  `rate` decimal(18,8) NOT NULL,

  `date` date NOT NULL,

  PRIMARY KEY (`from\_currency`,`to\_currency`,`date`),

  KEY `to\_currency` (`to\_currency`),

  CONSTRAINT `exchange\_rates\_ibfk\_1` FOREIGN KEY (`from\_currency`) REFERENCES `currencies` (`code`) ON DELETE CASCADE,

  CONSTRAINT `exchange\_rates\_ibfk\_2` FOREIGN KEY (`to\_currency`) REFERENCES `currencies` (`code`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `exchange\_rates` (`from\_currency`, `to\_currency`, `rate`, `date`) VALUES

  ('EUR', 'USD', 1.18000000, '2025-04-01'),

  ('GBP', 'USD', 1.33000000, '2025-04-01'),

  ('USD', 'EUR', 0.85000000, '2025-04-01'),

  ('USD', 'GBP', 0.75000000, '2025-04-01');

CREATE TABLE IF NOT EXISTS `expenses` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `employee\_id` int(11) NOT NULL,

  `expense\_category\_id` int(11) NOT NULL,

  `amount` decimal(10,2) NOT NULL,

  `incurred\_on` date NOT NULL,

  PRIMARY KEY (`id`),

  KEY `employee\_id` (`employee\_id`),

  KEY `expense\_category\_id` (`expense\_category\_id`),

  CONSTRAINT `expenses\_ibfk\_1` FOREIGN KEY (`employee\_id`) REFERENCES `employees` (`id`) ON DELETE CASCADE,

  CONSTRAINT `expenses\_ibfk\_2` FOREIGN KEY (`expense\_category\_id`) REFERENCES `expense\_categories` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `expenses` (`id`, `employee\_id`, `expense\_category\_id`, `amount`, `incurred\_on`) VALUES

  (1, 1, 1, 100.00, '2025-02-10'),

  (2, 2, 2, 50.00, '2025-02-11'),

  (3, 3, 3, 25.00, '2025-02-12');

CREATE TABLE IF NOT EXISTS `expense\_categories` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `expense\_categories` (`id`, `name`) VALUES

  (2, 'Meals'),

  (3, 'Supplies'),

  (1, 'Travel');

DELIMITER //

CREATE FUNCTION `fn\_calculate\_order\_total`(p\_order\_id INT) RETURNS decimal(12,2)

    DETERMINISTIC

BEGIN

  DECLARE total DECIMAL(12,2);

  SELECT SUM(quantity \* unit\_price) INTO total FROM `order\_items` WHERE order\_id = p\_order\_id;

  RETURN IFNULL(total, 0);

END//

DELIMITER ;

DELIMITER //

CREATE FUNCTION `fn\_calculate\_tax`(p\_amount DECIMAL(12,2), p\_tax\_rate DECIMAL(5,4)) RETURNS decimal(12,2)

    DETERMINISTIC

RETURN ROUND(p\_amount \* p\_tax\_rate, 2)//

DELIMITER ;

DELIMITER //

CREATE FUNCTION `fn\_days\_between`(p\_start DATE, p\_end DATE) RETURNS int(11)

    DETERMINISTIC

RETURN DATEDIFF(p\_end, p\_start)//

DELIMITER ;

DELIMITER //

CREATE FUNCTION `fn\_format\_currency`(p\_amount DECIMAL(12,2), p\_symbol VARCHAR(10)) RETURNS varchar(50) CHARSET utf8mb4 COLLATE utf8mb4\_unicode\_ci

    DETERMINISTIC

RETURN CONCAT(p\_symbol, FORMAT(p\_amount, 2))//

DELIMITER ;

DELIMITER //

CREATE FUNCTION `fn\_get\_customer\_balance`(p\_customer\_id INT) RETURNS decimal(12,2)

    DETERMINISTIC

BEGIN

  DECLARE bal DECIMAL(12,2);

  SELECT IFNULL(SUM(i.total\_amount) - SUM(IFNULL(p.amount, 0)), 0)

  INTO bal

  FROM `invoices` i

  LEFT JOIN `payments` p ON i.id = p.invoice\_id

  WHERE i.order\_id IN (SELECT id FROM `orders` WHERE customer\_id = p\_customer\_id);

  RETURN bal;

END//

DELIMITER ;

DELIMITER //

CREATE FUNCTION `fn\_get\_employee\_fullname`(p\_emp\_id INT) RETURNS varchar(101) CHARSET utf8mb4 COLLATE utf8mb4\_unicode\_ci

    DETERMINISTIC

BEGIN

  DECLARE fname VARCHAR(50);

  DECLARE lname VARCHAR(50);

  SELECT first\_name, last\_name INTO fname, lname FROM `employees` WHERE id = p\_emp\_id;

  RETURN CONCAT(fname, ' ', lname);

END//

DELIMITER ;

DELIMITER //

CREATE FUNCTION `fn\_get\_stock\_level`(p\_product\_id INT) RETURNS int(11)

    DETERMINISTIC

BEGIN

  DECLARE qty INT;

  SELECT SUM(quantity\_on\_hand) INTO qty FROM `inventory` WHERE product\_id = p\_product\_id;

  RETURN IFNULL(qty, 0);

END//

DELIMITER ;

CREATE TABLE IF NOT EXISTS `inventory` (

  `product\_id` int(11) NOT NULL,

  `warehouse\_id` int(11) NOT NULL,

  `quantity\_on\_hand` int(11) NOT NULL DEFAULT 0,

  PRIMARY KEY (`product\_id`,`warehouse\_id`),

  KEY `warehouse\_id` (`warehouse\_id`),

  CONSTRAINT `inventory\_ibfk\_1` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE CASCADE,

  CONSTRAINT `inventory\_ibfk\_2` FOREIGN KEY (`warehouse\_id`) REFERENCES `warehouses` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `inventory` (`product\_id`, `warehouse\_id`, `quantity\_on\_hand`) VALUES

  (1, 1, 180),

  (1, 2, 200),

  (2, 1, 250),

  (3, 1, 70),

  (3, 3, 300),

  (4, 1, 40),

  (4, 2, 40),

  (4, 4, 400);

CREATE TABLE IF NOT EXISTS `invoices` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `order\_id` int(11) NOT NULL,

  `invoice\_date` date NOT NULL,

  `due\_date` date NOT NULL,

  `total\_amount` decimal(12,2) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `order\_id` (`order\_id`),

  CONSTRAINT `invoices\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `invoices` (`id`, `order\_id`, `invoice\_date`, `due\_date`, `total\_amount`, `created\_at`) VALUES

  (1, 1, '2025-02-05', '2025-02-15', 40.00, '2025-02-05 09:00:00'),

  (2, 2, '2025-02-06', '2025-02-16', 90.00, '2025-02-06 09:00:00'),

  (3, 3, '2025-02-07', '2025-02-17', 100.00, '2025-02-07 09:00:00'),

  (5, 5, '2025-05-14', '2025-06-13', 200.00, '2025-05-14 07:14:50');

CREATE TABLE IF NOT EXISTS `invoice\_items` (

  `invoice\_id` int(11) NOT NULL,

  `order\_item\_order\_id` int(11) NOT NULL,

  `order\_item\_product\_id` int(11) NOT NULL,

  `quantity` int(11) NOT NULL,

  `unit\_price` decimal(10,2) NOT NULL,

  PRIMARY KEY (`invoice\_id`,`order\_item\_order\_id`,`order\_item\_product\_id`),

  KEY `order\_item\_order\_id` (`order\_item\_order\_id`,`order\_item\_product\_id`),

  CONSTRAINT `invoice\_items\_ibfk\_1` FOREIGN KEY (`invoice\_id`) REFERENCES `invoices` (`id`) ON DELETE CASCADE,

  CONSTRAINT `invoice\_items\_ibfk\_2` FOREIGN KEY (`order\_item\_order\_id`, `order\_item\_product\_id`) REFERENCES `order\_items` (`order\_id`, `product\_id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `invoice\_items` (`invoice\_id`, `order\_item\_order\_id`, `order\_item\_product\_id`, `quantity`, `unit\_price`) VALUES

  (1, 1, 1, 2, 10.00),

  (1, 1, 2, 1, 20.00),

  (2, 2, 2, 3, 20.00),

  (2, 2, 3, 1, 30.00),

  (3, 3, 3, 2, 30.00),

  (3, 3, 4, 1, 40.00);

CREATE TABLE IF NOT EXISTS `logs` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `user\_id` int(11) DEFAULT NULL,

  `action` varchar(100) NOT NULL,

  `entity` varchar(100) DEFAULT NULL,

  `entity\_id` int(11) DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `user\_id` (`user\_id`),

  CONSTRAINT `logs\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `logs` (`id`, `user\_id`, `action`, `entity`, `entity\_id`, `created\_at`) VALUES

  (1, NULL, 'Payment of 40.00 recorded for invoice 1', 'payments', 1, '2025-05-14 05:57:58'),

  (2, NULL, 'Payment of 90.00 recorded for invoice 2', 'payments', 2, '2025-05-14 05:57:58'),

  (3, NULL, 'Payment of 100.00 recorded for invoice 3', 'payments', 3, '2025-05-14 05:57:58'),

  (4, NULL, 'Payment of 90.00 recorded for invoice 4', 'payments', 4, '2025-05-14 05:57:58'),

  (5, 1, 'LOGIN', 'sessions', 1, '2025-01-01 06:05:00'),

  (6, 2, 'CREATE', 'orders', 2, '2025-02-02 09:05:00'),

  (7, NULL, 'Order status changed from New to Completed', 'orders', 5, '2025-05-14 07:21:47');

CREATE TABLE IF NOT EXISTS `maintenance\_logs` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `request\_id` int(11) NOT NULL,

  `logged\_on` datetime NOT NULL,

  `notes` text DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `request\_id` (`request\_id`),

  CONSTRAINT `maintenance\_logs\_ibfk\_1` FOREIGN KEY (`request\_id`) REFERENCES `maintenance\_requests` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `maintenance\_logs` (`id`, `request\_id`, `logged\_on`, `notes`) VALUES

  (1, 1, '2025-02-03 10:00:00', 'Fixed issue'),

  (2, 2, '2025-02-04 11:00:00', 'Replaced part');

CREATE TABLE IF NOT EXISTS `maintenance\_requests` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `asset\_id` int(11) NOT NULL,

  `requested\_by` int(11) DEFAULT NULL,

  `requested\_on` datetime NOT NULL,

  `status` varchar(50) DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `asset\_id` (`asset\_id`),

  KEY `requested\_by` (`requested\_by`),

  CONSTRAINT `maintenance\_requests\_ibfk\_1` FOREIGN KEY (`asset\_id`) REFERENCES `assets` (`id`) ON DELETE CASCADE,

  CONSTRAINT `maintenance\_requests\_ibfk\_2` FOREIGN KEY (`requested\_by`) REFERENCES `employees` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `maintenance\_requests` (`id`, `asset\_id`, `requested\_by`, `requested\_on`, `status`) VALUES

  (1, 1, 1, '2025-02-01 08:00:00', 'Open'),

  (2, 2, 2, '2025-02-02 09:00:00', 'Closed');

CREATE TABLE IF NOT EXISTS `message\_templates` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `subject` varchar(255) DEFAULT NULL,

  `body` text DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS `notifications` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `title` varchar(100) NOT NULL,

  `body` text NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `notifications` (`id`, `title`, `body`, `created\_at`) VALUES

  (1, 'Welcome', 'Welcome to the system', '2025-01-01 06:00:00'),

  (2, 'Reminder', 'Your invoice is due', '2025-02-10 06:00:00');

CREATE TABLE IF NOT EXISTS `notification\_recipients` (

  `notification\_id` int(11) NOT NULL,

  `user\_id` int(11) NOT NULL,

  `read` tinyint(1) DEFAULT 0,

  PRIMARY KEY (`notification\_id`,`user\_id`),

  KEY `user\_id` (`user\_id`),

  CONSTRAINT `notification\_recipients\_ibfk\_1` FOREIGN KEY (`notification\_id`) REFERENCES `notifications` (`id`) ON DELETE CASCADE,

  CONSTRAINT `notification\_recipients\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `notification\_recipients` (`notification\_id`, `user\_id`, `read`) VALUES

  (1, 1, 0),

  (1, 2, 1),

  (2, 1, 0),

  (2, 3, 0);

CREATE TABLE IF NOT EXISTS `orders` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `customer\_id` int(11) NOT NULL,

  `order\_date` date NOT NULL,

  `status` varchar(50) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `customer\_id` (`customer\_id`),

  CONSTRAINT `orders\_ibfk\_1` FOREIGN KEY (`customer\_id`) REFERENCES `customers` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `orders` (`id`, `customer\_id`, `order\_date`, `status`, `created\_at`) VALUES

  (1, 1, '2025-02-01', 'Pending', '2025-02-01 09:00:00'),

  (2, 2, '2025-02-02', 'Shipped', '2025-02-02 09:00:00'),

  (3, 3, '2025-02-03', 'Completed', '2025-02-03 09:00:00'),

  (5, 1, '2025-05-10', 'Completed', '2025-05-14 06:56:17');

CREATE TABLE IF NOT EXISTS `order\_items` (

  `order\_id` int(11) NOT NULL,

  `product\_id` int(11) NOT NULL,

  `quantity` int(11) NOT NULL,

  `unit\_price` decimal(10,2) NOT NULL,

  PRIMARY KEY (`order\_id`,`product\_id`),

  KEY `product\_id` (`product\_id`),

  CONSTRAINT `order\_items\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE,

  CONSTRAINT `order\_items\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `order\_items` (`order\_id`, `product\_id`, `quantity`, `unit\_price`) VALUES

  (1, 1, 2, 10.00),

  (1, 2, 1, 20.00),

  (2, 2, 3, 20.00),

  (2, 3, 1, 30.00),

  (3, 3, 2, 30.00),

  (3, 4, 1, 40.00),

  (5, 2, 10, 20.00);

CREATE TABLE IF NOT EXISTS `organizations` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `address` varchar(255) DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `organizations` (`id`, `name`, `address`, `created\_at`) VALUES

  (1, 'Org A', '123 Main St', '2025-01-01 06:00:00'),

  (2, 'Org B', '456 Elm St', '2025-01-02 06:00:00'),

  (3, 'Org C', '789 Oak St', '2025-01-03 06:00:00'),

  (4, 'Org D', '321 Pine St', '2025-01-04 06:00:00');

CREATE TABLE IF NOT EXISTS `payments` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `invoice\_id` int(11) NOT NULL,

  `payment\_date` date NOT NULL,

  `amount` decimal(12,2) NOT NULL,

  `payment\_method\_id` int(11) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `invoice\_id` (`invoice\_id`),

  KEY `payment\_method\_id` (`payment\_method\_id`),

  CONSTRAINT `payments\_ibfk\_1` FOREIGN KEY (`invoice\_id`) REFERENCES `invoices` (`id`) ON DELETE CASCADE,

  CONSTRAINT `payments\_ibfk\_2` FOREIGN KEY (`payment\_method\_id`) REFERENCES `payment\_methods` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `payments` (`id`, `invoice\_id`, `payment\_date`, `amount`, `payment\_method\_id`, `created\_at`) VALUES

  (1, 1, '2025-02-10', 40.00, 1, '2025-02-10 10:00:00'),

  (2, 2, '2025-02-11', 90.00, 2, '2025-02-11 10:00:00'),

  (3, 3, '2025-02-12', 100.00, 3, '2025-02-12 10:00:00');

CREATE TABLE IF NOT EXISTS `payment\_methods` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `method\_name` varchar(50) NOT NULL,

  `details` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `method\_name` (`method\_name`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `payment\_methods` (`id`, `method\_name`, `details`) VALUES

  (1, 'Credit Card', 'Visa'),

  (2, 'PayPal', 'PayPal account'),

  (3, 'Bank Transfer', 'IBAN details');

CREATE TABLE IF NOT EXISTS `permissions` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `description` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `permissions` (`id`, `name`, `description`) VALUES

  (1, 'read', 'Read permission'),

  (2, 'write', 'Write permission'),

  (3, 'delete', 'Delete permission'),

  (4, 'update', 'Update permission');

CREATE TABLE IF NOT EXISTS `price\_lists` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `effective\_date` date NOT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `price\_lists` (`id`, `name`, `effective\_date`) VALUES

  (1, 'PriceList A', '2025-01-01'),

  (2, 'PriceList B', '2025-02-01'),

  (3, 'PriceList C', '2025-03-01');

CREATE TABLE IF NOT EXISTS `price\_list\_items` (

  `price\_list\_id` int(11) NOT NULL,

  `product\_id` int(11) NOT NULL,

  `price` decimal(10,2) NOT NULL,

  PRIMARY KEY (`price\_list\_id`,`product\_id`),

  KEY `product\_id` (`product\_id`),

  CONSTRAINT `price\_list\_items\_ibfk\_1` FOREIGN KEY (`price\_list\_id`) REFERENCES `price\_lists` (`id`) ON DELETE CASCADE,

  CONSTRAINT `price\_list\_items\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `price\_list\_items` (`price\_list\_id`, `product\_id`, `price`) VALUES

  (1, 1, 9.50),

  (1, 2, 19.00),

  (2, 3, 29.00),

  (2, 4, 39.00),

  (3, 1, 9.00),

  (3, 4, 38.00);

CREATE TABLE IF NOT EXISTS `products` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `supplier\_id` int(11) DEFAULT NULL,

  `category\_id` int(11) DEFAULT NULL,

  `sku` varchar(50) NOT NULL,

  `name` varchar(100) NOT NULL,

  `description` text DEFAULT NULL,

  `unit\_price` decimal(10,2) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  UNIQUE KEY `sku` (`sku`),

  KEY `supplier\_id` (`supplier\_id`),

  KEY `category\_id` (`category\_id`),

  CONSTRAINT `products\_ibfk\_1` FOREIGN KEY (`supplier\_id`) REFERENCES `suppliers` (`id`) ON DELETE CASCADE,

  CONSTRAINT `products\_ibfk\_2` FOREIGN KEY (`category\_id`) REFERENCES `categories` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `products` (`id`, `supplier\_id`, `category\_id`, `sku`, `name`, `description`, `unit\_price`, `created\_at`) VALUES

  (1, 1, 1, 'SKU001', 'Product A', 'Desc A', 10.00, '2025-01-05 08:00:00'),

  (2, 2, 2, 'SKU002', 'Product B', 'Desc B', 20.00, '2025-01-06 08:00:00'),

  (3, 3, 3, 'SKU003', 'Product C', 'Desc C', 30.00, '2025-01-07 08:00:00'),

  (4, 4, 4, 'SKU004', 'Product D', 'Desc D', 40.00, '2025-01-08 08:00:00');

CREATE TABLE IF NOT EXISTS `product\_categories` (

  `product\_id` int(11) NOT NULL,

  `category\_id` int(11) NOT NULL,

  PRIMARY KEY (`product\_id`,`category\_id`),

  KEY `category\_id` (`category\_id`),

  CONSTRAINT `product\_categories\_ibfk\_1` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE CASCADE,

  CONSTRAINT `product\_categories\_ibfk\_2` FOREIGN KEY (`category\_id`) REFERENCES `categories` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `product\_categories` (`product\_id`, `category\_id`) VALUES

  (1, 2),

  (1, 3),

  (2, 1),

  (3, 4),

  (4, 1),

  (4, 2);

CREATE TABLE IF NOT EXISTS `projects` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `start\_date` date DEFAULT NULL,

  `end\_date` date DEFAULT NULL,

  `status` varchar(50) DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `projects` (`id`, `name`, `start\_date`, `end\_date`, `status`) VALUES

  (1, 'Project Alpha', '2025-01-01', '2025-06-01', 'Active'),

  (2, 'Project Beta', '2025-02-01', '2025-07-01', 'Planned');

CREATE TABLE IF NOT EXISTS `project\_tasks` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `project\_id` int(11) NOT NULL,

  `name` varchar(100) NOT NULL,

  `assignee\_id` int(11) DEFAULT NULL,

  `due\_date` date DEFAULT NULL,

  `status` varchar(50) DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `project\_id` (`project\_id`),

  KEY `assignee\_id` (`assignee\_id`),

  CONSTRAINT `project\_tasks\_ibfk\_1` FOREIGN KEY (`project\_id`) REFERENCES `projects` (`id`) ON DELETE CASCADE,

  CONSTRAINT `project\_tasks\_ibfk\_2` FOREIGN KEY (`assignee\_id`) REFERENCES `employees` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `project\_tasks` (`id`, `project\_id`, `name`, `assignee\_id`, `due\_date`, `status`) VALUES

  (1, 1, 'Task 1', 1, '2025-03-01', 'Open'),

  (2, 1, 'Task 2', 2, '2025-04-01', 'InProgress'),

  (3, 2, 'Task 3', 3, '2025-05-01', 'Open');

CREATE TABLE IF NOT EXISTS `purchase\_orders` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `supplier\_id` int(11) NOT NULL,

  `order\_date` date NOT NULL,

  `status` varchar(50) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `supplier\_id` (`supplier\_id`),

  CONSTRAINT `purchase\_orders\_ibfk\_1` FOREIGN KEY (`supplier\_id`) REFERENCES `suppliers` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `purchase\_orders` (`id`, `supplier\_id`, `order\_date`, `status`, `created\_at`) VALUES

  (1, 1, '2025-03-01', 'Ordered', '2025-03-01 07:00:00'),

  (2, 2, '2025-03-02', 'Received', '2025-03-02 07:00:00'),

  (3, 3, '2025-03-03', 'Cancelled', '2025-03-03 07:00:00'),

  (4, 4, '2025-03-04', 'Pending', '2025-03-04 07:00:00');

CREATE TABLE IF NOT EXISTS `purchase\_order\_items` (

  `purchase\_order\_id` int(11) NOT NULL,

  `product\_id` int(11) NOT NULL,

  `quantity` int(11) NOT NULL,

  `unit\_cost` decimal(10,2) NOT NULL,

  PRIMARY KEY (`purchase\_order\_id`,`product\_id`),

  KEY `product\_id` (`product\_id`),

  CONSTRAINT `purchase\_order\_items\_ibfk\_1` FOREIGN KEY (`purchase\_order\_id`) REFERENCES `purchase\_orders` (`id`) ON DELETE CASCADE,

  CONSTRAINT `purchase\_order\_items\_ibfk\_2` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `purchase\_order\_items` (`purchase\_order\_id`, `product\_id`, `quantity`, `unit\_cost`) VALUES

  (1, 1, 50, 8.00),

  (1, 2, 60, 15.00),

  (2, 3, 70, 25.00),

  (2, 4, 80, 35.00),

  (3, 1, 30, 8.00),

  (4, 2, 40, 15.00);

CREATE TABLE IF NOT EXISTS `roles` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(50) NOT NULL,

  `description` varchar(255) DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  UNIQUE KEY `name` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `roles` (`id`, `name`, `description`, `created\_at`) VALUES

  (1, 'Admin', 'Administrator role', '2025-01-01 06:00:00'),

  (2, 'Manager', 'Manager role', '2025-01-02 06:00:00'),

  (3, 'User', 'Regular user role', '2025-01-03 06:00:00'),

  (4, 'Guest', 'Guest role', '2025-01-04 06:00:00');

CREATE TABLE IF NOT EXISTS `role\_permissions` (

  `role\_id` int(11) NOT NULL,

  `permission\_id` int(11) NOT NULL,

  PRIMARY KEY (`role\_id`,`permission\_id`),

  KEY `permission\_id` (`permission\_id`),

  CONSTRAINT `role\_permissions\_ibfk\_1` FOREIGN KEY (`role\_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE,

  CONSTRAINT `role\_permissions\_ibfk\_2` FOREIGN KEY (`permission\_id`) REFERENCES `permissions` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `role\_permissions` (`role\_id`, `permission\_id`) VALUES

  (1, 1),

  (1, 2),

  (1, 3),

  (1, 4),

  (2, 1),

  (2, 2),

  (2, 4),

  (3, 1),

  (4, 1);

CREATE TABLE IF NOT EXISTS `sessions` (

  `id` char(128) NOT NULL,

  `user\_id` int(11) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  `expires\_at` timestamp NOT NULL,

  PRIMARY KEY (`id`),

  KEY `user\_id` (`user\_id`),

  CONSTRAINT `sessions\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `sessions` (`id`, `user\_id`, `created\_at`, `expires\_at`) VALUES

  ('sess1', 1, '2025-05-01 05:00:00', '2025-05-02 05:00:00'),

  ('sess2', 2, '2025-05-02 06:00:00', '2025-05-03 06:00:00');

CREATE TABLE IF NOT EXISTS `shipments` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `order\_id` int(11) NOT NULL,

  `shipment\_date` date NOT NULL,

  `carrier` varchar(100) DEFAULT NULL,

  `tracking\_number` varchar(100) DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`),

  KEY `order\_id` (`order\_id`),

  CONSTRAINT `shipments\_ibfk\_1` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `shipments` (`id`, `order\_id`, `shipment\_date`, `carrier`, `tracking\_number`, `created\_at`) VALUES

  (1, 1, '2025-02-03', 'UPS', '1Z999AA101', '2025-05-14 05:57:58'),

  (2, 2, '2025-02-04', 'FedEx', '999999999', '2025-05-14 05:57:58'),

  (3, 3, '2025-02-05', 'DHL', 'JVGL999', '2025-05-14 05:57:58');

CREATE TABLE IF NOT EXISTS `shipment\_items` (

  `shipment\_id` int(11) NOT NULL,

  `order\_item\_order\_id` int(11) NOT NULL,

  `order\_item\_product\_id` int(11) NOT NULL,

  `quantity` int(11) NOT NULL,

  PRIMARY KEY (`shipment\_id`,`order\_item\_order\_id`,`order\_item\_product\_id`),

  KEY `order\_item\_order\_id` (`order\_item\_order\_id`,`order\_item\_product\_id`),

  CONSTRAINT `shipment\_items\_ibfk\_1` FOREIGN KEY (`shipment\_id`) REFERENCES `shipments` (`id`) ON DELETE CASCADE,

  CONSTRAINT `shipment\_items\_ibfk\_2` FOREIGN KEY (`order\_item\_order\_id`, `order\_item\_product\_id`) REFERENCES `order\_items` (`order\_id`, `product\_id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `shipment\_items` (`shipment\_id`, `order\_item\_order\_id`, `order\_item\_product\_id`, `quantity`) VALUES

  (1, 1, 1, 2),

  (1, 1, 2, 1),

  (2, 2, 2, 3),

  (2, 2, 3, 1),

  (3, 3, 3, 2),

  (3, 3, 4, 1);

DELIMITER //

CREATE PROCEDURE `sp\_add\_product\_to\_order`(

  IN p\_order\_id INT,

  IN p\_product\_id INT,

  IN p\_quantity INT

)

BEGIN

  DECLARE price DECIMAL(10,2);

  SELECT unit\_price INTO price

  FROM products

  WHERE id = p\_product\_id

  LIMIT 1;

  IF price IS NULL THEN

    SIGNAL SQLSTATE '45000'

    SET MESSAGE\_TEXT = 'Product not found or unit\_price is NULL';

  END IF;

  INSERT INTO order\_items (order\_id, product\_id, quantity, unit\_price)

  VALUES (p\_order\_id, p\_product\_id, p\_quantity, price);

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE `sp\_assign\_employee\_to\_department`(

  IN p\_employee\_id INT,

  IN p\_department\_id INT

)

BEGIN

  INSERT IGNORE INTO `employee\_departments`(`employee\_id`,`department\_id`)

  VALUES(p\_employee\_id, p\_department\_id);

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE `sp\_create\_invoice`(

  IN p\_order\_id INT,

  OUT p\_invoice\_id INT

)

BEGIN

  DECLARE total DECIMAL(12,2) DEFAULT 0;

  SELECT SUM(quantity \* unit\_price) INTO total FROM `order\_items` WHERE order\_id = p\_order\_id;

  INSERT INTO `invoices`(`order\_id`,`invoice\_date`,`due\_date`,`total\_amount`)

  VALUES(p\_order\_id, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 30 DAY), total);

  SET p\_invoice\_id = LAST\_INSERT\_ID();

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE `sp\_create\_order`(

  IN p\_customer\_id INT,

  IN p\_order\_date DATE,

  OUT p\_order\_id INT

)

BEGIN

  INSERT INTO `orders`(`customer\_id`,`order\_date`,`status`)

  VALUES(p\_customer\_id, p\_order\_date, 'New');

  SET p\_order\_id = LAST\_INSERT\_ID();

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE `sp\_record\_payment`(

  IN p\_invoice\_id INT,

  IN p\_amount DECIMAL(12,2),

  IN p\_method\_id INT

)

BEGIN

  INSERT INTO `payments`(`invoice\_id`,`payment\_date`,`amount`,`payment\_method\_id`)

  VALUES(p\_invoice\_id, CURDATE(), p\_amount, p\_method\_id);

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE `sp\_transfer\_stock`(

  IN p\_product\_id INT,

  IN p\_from\_wh INT,

  IN p\_to\_wh INT,

  IN p\_quantity INT

)

BEGIN

  UPDATE `inventory` SET quantity\_on\_hand = quantity\_on\_hand - p\_quantity

    WHERE product\_id = p\_product\_id AND warehouse\_id = p\_from\_wh;

  INSERT INTO `inventory`(`product\_id`,`warehouse\_id`,`quantity\_on\_hand`)

    VALUES(p\_product\_id, p\_to\_wh, p\_quantity)

    ON DUPLICATE KEY UPDATE quantity\_on\_hand = quantity\_on\_hand + p\_quantity;

END//

DELIMITER ;

CREATE TABLE IF NOT EXISTS `suppliers` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `company\_name` varchar(100) NOT NULL,

  `contact\_name` varchar(100) DEFAULT NULL,

  `contact\_email` varchar(100) DEFAULT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `suppliers` (`id`, `company\_name`, `contact\_name`, `contact\_email`, `created\_at`) VALUES

  (1, 'Supplier A', 'Sam Supplier', 'sam.sup@example.com', '2025-01-05 07:00:00'),

  (2, 'Supplier B', 'Sue Supplier', 'sue.sup@example.com', '2025-01-06 07:00:00'),

  (3, 'Supplier C', 'Sid Supplier', 'sid.sup@example.com', '2025-01-07 07:00:00'),

  (4, 'Supplier D', 'Sky Supplier', 'sky.sup@example.com', '2025-01-08 07:00:00');

CREATE TABLE IF NOT EXISTS `system\_settings` (

  `key` varchar(100) NOT NULL,

  `value` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`key`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

CREATE TABLE IF NOT EXISTS `taxation\_rules` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `rate` decimal(5,4) NOT NULL,

  `applicable\_from` date DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `taxation\_rules` (`id`, `name`, `rate`, `applicable\_from`) VALUES

  (1, 'VAT', 0.2000, '2025-01-01'),

  (2, 'GST', 0.1000, '2025-01-15');

CREATE TABLE IF NOT EXISTS `timesheets` (

  `employee\_id` int(11) NOT NULL,

  `project\_task\_id` int(11) NOT NULL,

  `date` date NOT NULL,

  `hours` decimal(4,2) NOT NULL,

  PRIMARY KEY (`employee\_id`,`project\_task\_id`,`date`),

  KEY `project\_task\_id` (`project\_task\_id`),

  CONSTRAINT `timesheets\_ibfk\_1` FOREIGN KEY (`employee\_id`) REFERENCES `employees` (`id`) ON DELETE CASCADE,

  CONSTRAINT `timesheets\_ibfk\_2` FOREIGN KEY (`project\_task\_id`) REFERENCES `project\_tasks` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `timesheets` (`employee\_id`, `project\_task\_id`, `date`, `hours`) VALUES

  (1, 1, '2025-02-01', 8.00),

  (2, 1, '2025-02-01', 6.00),

  (2, 2, '2025-02-02', 7.50),

  (3, 3, '2025-02-03', 8.00);

CREATE TABLE IF NOT EXISTS `users` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `username` varchar(50) NOT NULL,

  `password\_hash` varchar(255) NOT NULL,

  `email` varchar(100) NOT NULL,

  `created\_at` timestamp NULL DEFAULT current\_timestamp(),

  `updated\_at` timestamp NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp(),

  PRIMARY KEY (`id`),

  UNIQUE KEY `username` (`username`),

  UNIQUE KEY `email` (`email`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `users` (`id`, `username`, `password\_hash`, `email`, `created\_at`, `updated\_at`) VALUES

  (1, 'alice', 'hash1', 'alice@example.com', '2025-01-01 06:00:00', '2025-01-01 06:00:00'),

  (2, 'bob', 'hash2', 'bob@example.com', '2025-01-02 07:00:00', '2025-01-02 07:00:00'),

  (3, 'carol', 'hash3', 'carol@example.com', '2025-01-03 08:00:00', '2025-01-03 08:00:00'),

  (4, 'dave', 'hash4', 'dave@example.com', '2025-01-04 09:00:00', '2025-01-04 09:00:00'),

  (5, 'test', 'test', 'test@test.com', '2025-05-13 21:00:00', '2025-05-13 21:00:00');

CREATE TABLE IF NOT EXISTS `user\_roles` (

  `user\_id` int(11) NOT NULL,

  `role\_id` int(11) NOT NULL,

  PRIMARY KEY (`user\_id`,`role\_id`),

  KEY `role\_id` (`role\_id`),

  CONSTRAINT `user\_roles\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`) ON DELETE CASCADE,

  CONSTRAINT `user\_roles\_ibfk\_2` FOREIGN KEY (`role\_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `user\_roles` (`user\_id`, `role\_id`) VALUES

  (1, 1),

  (2, 2),

  (3, 3),

  (4, 4);

CREATE TABLE `v\_customer\_balances` (

  `customer\_id` INT(11) NOT NULL,

  `company\_name` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `balance` DECIMAL(12,2) NULL

) ENGINE=MyISAM;

CREATE TABLE `v\_employee\_directory` (

  `employee\_id` INT(11) NOT NULL,

  `fullname` VARCHAR(1) NULL COLLATE 'utf8mb4\_unicode\_ci',

  `email` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `phone` VARCHAR(1) NULL COLLATE 'utf8mb4\_unicode\_ci'

) ENGINE=MyISAM;

CREATE TABLE `v\_inventory\_levels` (

  `product\_id` INT(11) NOT NULL,

  `name` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `quantity\_on\_hand` INT(11) NULL

) ENGINE=MyISAM;

CREATE TABLE `v\_order\_overview` (

  `order\_id` INT(11) NOT NULL,

  `company\_name` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `order\_date` DATE NOT NULL,

  `status` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `total\_amount` DECIMAL(12,2) NULL

) ENGINE=MyISAM;

CREATE TABLE `v\_purchase\_order\_status` (

  `id` INT(11) NOT NULL,

  `supplier` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `order\_date` DATE NOT NULL,

  `status` VARCHAR(1) NOT NULL COLLATE 'utf8mb4\_unicode\_ci',

  `total\_cost` DECIMAL(42,2) NULL

) ENGINE=MyISAM;

CREATE TABLE `v\_sales\_summary` (

  `sale\_date` DATE NULL,

  `daily\_sales` DECIMAL(42,2) NULL

) ENGINE=MyISAM;

CREATE TABLE IF NOT EXISTS `warehouses` (

  `id` int(11) NOT NULL AUTO\_INCREMENT,

  `name` varchar(100) NOT NULL,

  `location` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

INSERT INTO `warehouses` (`id`, `name`, `location`) VALUES

  (1, 'Warehouse A', 'Location A'),

  (2, 'Warehouse B', 'Location B'),

  (3, 'Warehouse C', 'Location C'),

  (4, 'Warehouse D', 'Location D');

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_inventory\_after\_update`

AFTER UPDATE ON `inventory`

FOR EACH ROW

BEGIN

  INSERT INTO `audit\_trail`(`table\_name`,`record\_id`,`action`,`changed\_by`)

  VALUES('inventory', OLD.product\_id, 'UPDATE', NULL);

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_orders\_status\_change`

BEFORE UPDATE ON `orders`

FOR EACH ROW

BEGIN

  IF OLD.status <> NEW.status THEN

    INSERT INTO `logs`(`user\_id`,`action`,`entity`,`entity\_id`)

    VALUES(NULL, CONCAT('Order status changed from ', OLD.status, ' to ', NEW.status), 'orders', NEW.id);

  END IF;

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_payments\_after\_insert`

AFTER INSERT ON `payments`

FOR EACH ROW

BEGIN

  INSERT INTO `logs`(`user\_id`,`action`,`entity`,`entity\_id`)

  VALUES(NULL, CONCAT('Payment of ', NEW.amount, ' recorded for invoice ', NEW.invoice\_id), 'payments', NEW.id);

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_products\_before\_update`

BEFORE UPDATE ON `products`

FOR EACH ROW

BEGIN

  SET NEW.created\_at = OLD.created\_at;

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_purchase\_order\_after\_insert`

AFTER INSERT ON `purchase\_order\_items`

FOR EACH ROW

BEGIN

  INSERT INTO `inventory`(`product\_id`,`warehouse\_id`,`quantity\_on\_hand`)

  VALUES(NEW.product\_id, 1, NEW.quantity)

  ON DUPLICATE KEY UPDATE quantity\_on\_hand = quantity\_on\_hand + NEW.quantity;

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_shipments\_before\_insert`

BEFORE INSERT ON `shipments`

FOR EACH ROW

BEGIN

  SET NEW.created\_at = NOW();

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

SET @OLDTMP\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='STRICT\_TRANS\_TABLES,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_AUTO\_CREATE\_USER,NO\_ENGINE\_SUBSTITUTION';

DELIMITER //

CREATE TRIGGER `trg\_users\_after\_insert`

AFTER INSERT ON `users`

FOR EACH ROW

BEGIN

  INSERT INTO `audit\_trail`(`table\_name`,`record\_id`,`action`,`changed\_by`)

  VALUES('users', NEW.id, 'INSERT', NEW.id);

END//

DELIMITER ;

SET SQL\_MODE=@OLDTMP\_SQL\_MODE;

DROP TABLE IF EXISTS `v\_customer\_balances`;

CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `v\_customer\_balances` AS SELECT c.id AS customer\_id, c.company\_name,

       fn\_get\_customer\_balance(c.id) AS balance

FROM customers c;

DROP TABLE IF EXISTS `v\_employee\_directory`;

CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `v\_employee\_directory` AS SELECT id AS employee\_id, fn\_get\_employee\_fullname(id) AS fullname, email, phone

FROM employees;

DROP TABLE IF EXISTS `v\_inventory\_levels`;

CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `v\_inventory\_levels` AS SELECT p.id AS product\_id, p.name, fn\_get\_stock\_level(p.id) AS quantity\_on\_hand

FROM products p;

DROP TABLE IF EXISTS `v\_order\_overview`;

CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `v\_order\_overview` AS SELECT o.id AS order\_id, c.company\_name, o.order\_date, o.status,

       fn\_calculate\_order\_total(o.id) AS total\_amount

FROM orders o

JOIN customers c ON o.customer\_id = c.id;

DROP TABLE IF EXISTS `v\_purchase\_order\_status`;

CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `v\_purchase\_order\_status` AS SELECT po.id, s.company\_name AS supplier, po.order\_date, po.status,

       SUM(poi.quantity \* poi.unit\_cost) AS total\_cost

FROM purchase\_orders po

JOIN suppliers s ON po.supplier\_id = s.id

JOIN purchase\_order\_items poi ON po.id = poi.purchase\_order\_id

GROUP BY po.id;

DROP TABLE IF EXISTS `v\_sales\_summary`;

CREATE ALGORITHM=UNDEFINED SQL SECURITY DEFINER VIEW `v\_sales\_summary` AS SELECT DATE(o.order\_date) AS sale\_date,

       SUM(oi.quantity \* oi.unit\_price) AS daily\_sales

FROM orders o

JOIN order\_items oi ON o.id = oi.order\_id

GROUP BY DATE(o.order\_date);