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

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.

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

1.1. 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.

1.2. 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.

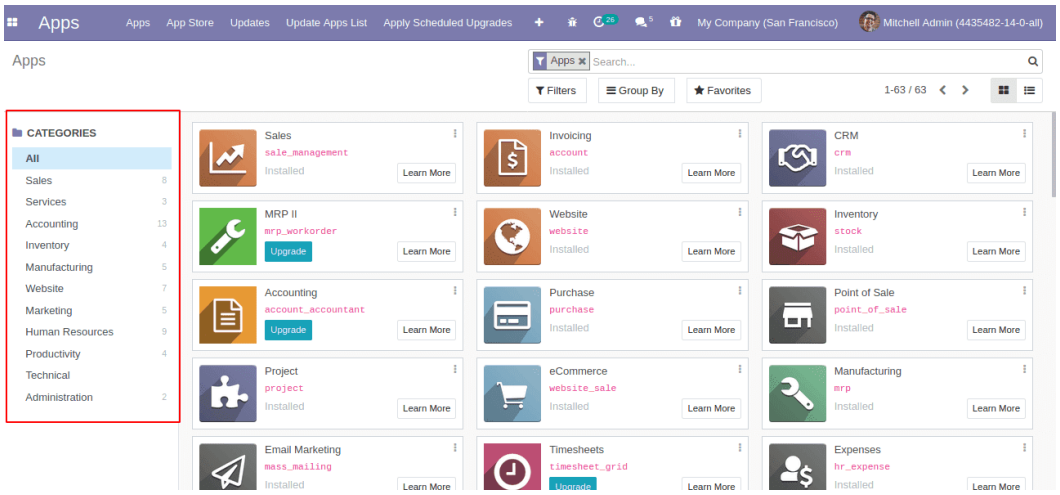

Рисунок 1.1 – Интерфейс Odoo

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.

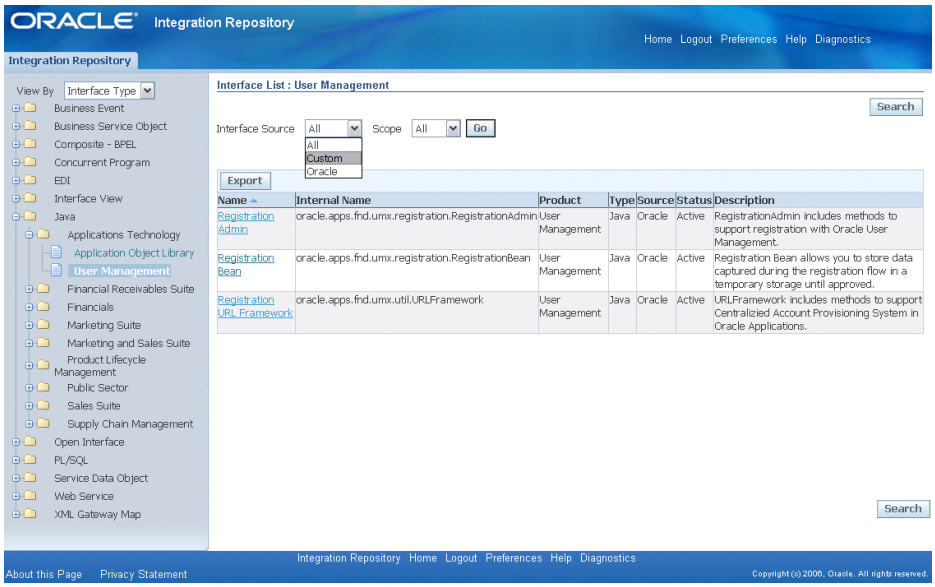

Рисунок 1.2 – Интерфейс Oracle E-Business Suite

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.

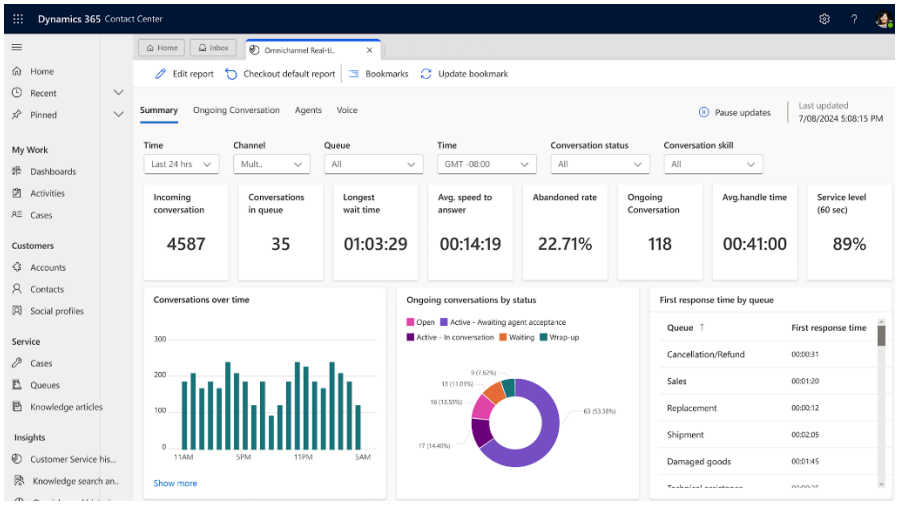

Рисунок 1.3 – Интерфейс Microsoft Dynamics 365

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.

1.3.Постановка задачи

1.3.1. 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.

1.3.2. Состав выполняемых функций

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

1. Организация работы администратора системы:
    a. Управление пользователями и ролями сотрудников;
    b. Контроль и аудит бизнес-процессов через систему логирования;
    c. Настройка справочников (единицы измерения, статусы, типы оплат и т.п.);
    d. Управление подразделениями и структурами предприятия;
    e. Назначение прав доступа к различным функциям системы;
    f. Просмотр аналитической информации и отчётности.
2. Организация работы менеджера по продажам:
    a. Создание и управление заказами клиентов;
    b. Добавление товаров в заказы и формирование счетов;
    c. Отслеживание статусов заказов;
    d. Проведение оплат и контроль задолженностей клиентов;
    e. Ведение клиентской базы.

3. Организация работы склада:
    a. Учёт остатков товаров на складах;
    b. Проведение инвентаризаций и перемещений между складами;
    c. Приёмка товаров по закупочным заказам;
    d. Формирование и обновление информации по наличию продукции.
4. Организация работы отдела закупок:
    a. Создание и управление закупочными заказами;
    b. Контроль поставок от поставщиков;
    c. Учёт стоимости закупок и прогнозирование потребностей;
    d. Ведение базы поставщиков.
5. Регистрация и авторизация сотрудников:
    a. Регистрация сотрудников различных ролей (администраторов, менеджеров, кладовщиков);
    b. Авторизация и контроль активности пользователей в системе.
6. Поддержка аналитики и отчетности:
    a. Отображение финансовых и складских отчётов;
    b. Формирование сводной информации по заказам, клиентам и поставщикам;
    c. Просмотр динамики продаж и остатков;
    d. Поддержка бизнес-решений через агрегированные представления данных.

1.3.3. 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.

1.3.4. 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.

1.3.5. Требования к составу параметрам технических и программных средств

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

1.3.6. Требования к информационной и программной совместимости

1. Информационная совместимость:
    a. Использование кодировки UTF-8 (utf8mb4) для обеспечения корректной работы с многоязычным контентом;
    b. Форматы хранения данных, совместимые с другими корпоративными системами (например, CSV, JSON, XML);
    c. Возможность масштабирования структуры базы данных под изменяющиеся бизнес-процессы.
2. Программная совместимость:
    a. Поддержка работы на основных платформах: Windows, Linux, macOS;

b. Возможность размещения в облачных инфраструктурах: AWS (Amazon RDS), Google Cloud SQL, Azure Database for MySQL, DigitalOcean Managed Databases;
c. Интеграция с внешними приложениями и сервисами через API и поддержка взаимодействия с языками программирования, такими как Java, C#, PHP, Python, JavaScript;
d. Совместимость с системами контроля версий (Git) и CI/CD-пайплайнами при развёртывании в корпоративной среде.

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1.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:
    a. Популярность и зрелость — MySQL является одной из самых широко используемых реляционных СУБД, что гарантирует стабильность и наличие большого количества обучающих материалов;
    b. Открытость и бесплатность — используется MySQL Community Edition, не требующая лицензирования;
    c. Производительность и масштабируемость — обеспечивает высокую производительность и возможность масштабирования при увеличении объёмов данных;
    d. Поддержка транзакций и ограничений целостности — позволяет надёжно управлять критичными бизнес-операциями в рамках ERP-системы.
2. Преимущества MySQL Workbench:
    a. Интуитивно понятный интерфейс для работы с таблицами, представлениями, индексами и пользователями;
    b. Удобное написание и отладка SQL-запросов благодаря встроенному редактору с подсветкой синтаксиса и автодополнением;
    c. Визуальное проектирование схемы БД — возможность создавать и изменять структуру данных с помощью диаграмм;
    d. Средства диагностики и администрирования — включают в себя мониторинг производительности и анализ нагрузок.
3. Преимущества dbdiagram.io:
    a. Простота создания ER-диаграмм в текстовом формате с мгновенной визуализацией;
    b. Быстрое редактирование схем без установки дополнительного ПО;
    c. Генерация SQL-кода по диаграммам для удобного импорта в СУБД;
    d. Возможность совместной работы — поддержка экспорта, импорта и публикации моделей.

2.2. 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:
    a. Пользователи системы — идентификаторы, контактные данные, роли и права доступа;
    b. Клиенты и поставщики — наименования, реквизиты, история заказов и оплат;
    c. Заказы и счета — статусы, даты оформления, суммы, связанные товары и услуги;
    d. Товары и складские запасы — названия, артикула, остатки, цены, единицы измерения;
    e. Сотрудники — ФИО, должности, контакты, активность в системе;
    f. Платежи и финансовые транзакции — суммы, даты, типы операций;
    g. Производственные и закупочные документы — спецификации, стоимости, статусы выполнения.
2. Учитываемые функции и действия:
    a. Управление заказами клиентов и поставками от поставщиков;
    b. Отслеживание остатков на складе и движения товаров;
    c. Расчёт общей стоимости заказов и остатков в режиме реального времени;
    d. Регистрация и логирование изменений статусов и ключевых действий;
    e. Учёт финансовых операций, включая выставление счетов и проведение платежей;
    f. Поддержка рабочих ролей: администратор, менеджер, кладовщик и бухгалтер.
3. Бизнес-процессы, моделируемые системой:
    a. Приём и обработка заказов клиентов;
    b. Закупка и приёмка товаров на склад;
    c. Выдача товаров и списание остатков;
    d. Управление статусами заказов и поставок;
    e. Ведение финансового учёта: оплата счетов, поступления и расходы;
    f. Ведение пользовательской и технической документации: логирование, аудит изменений;
    g. Подготовка аналитических данных — отчёты о продажах, остатках, задолженностях.

2.3. 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:
    a. Система должна хранить информацию о пользователях (ФИО, логин, email, статус и т.д.).
    b. Каждый пользователь должен иметь привязанную роль, определяющую его уровень доступа (например, администратор, сотрудник и т.д.).
    c. Пользователь может быть активным или деактивированным, что влияет на возможность входа в систему.
2. Управление документами:
    a. Система должна позволять хранить документы с основными атрибутами (название, описание, дата создания, тип документа и т.д.).
    b. Каждый документ должен иметь автора, связанного с конкретным пользователем.
    c. Документы должны классифицироваться по типу и быть связаны с определёнными тегами для облегчения поиска и фильтрации.
    d. Для документа необходимо отслеживать текущий статус (например, «в работе», «утверждён», «отклонён»).
3. Назначение ответственных лиц:
    a. Для каждого документа должен быть назначен один или несколько ответственных пользователей.
    b. Требуется фиксировать дату назначения и возможность замены ответственного.
4. Тегирование и классификация:
    a. Каждый документ может быть связан с одним или несколькими тегами, которые позволяют группировать и фильтровать документы по тематикам.
5. Уведомления
    a. Система должна создавать уведомления для пользователей о событиях (например, назначение ответственного, изменение статуса документа и т.д.).
    b. Уведомления должны содержать тип события, дату создания и ID связанного документа или пользователя.
6. Аудит действий:

a. Необходимо вести журнал действий пользователей (например, вход в систему, создание или редактирование документа, изменение статуса и т.д.).
b. Для каждой записи аудита фиксируются ID пользователя, тип действия, дата и связанный объект (например, документ или уведомление).

2.4.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.

2.5.Пользователи системы и их роли

Система предполагает участие следующих пользователей с их основными задачами:

1. Администратор:
    a. Имеет полный доступ ко всем сущностям и настройкам базы данных;
    b. Управляет пользователями, ролями и правами доступа;
    c. Настраивает справочники системы (категории товаров, статусы заказов, типы платежей и др.);
    d. Контролирует работу всех модулей и осуществляет аудит изменений.
2. Менеджер по продажам:
    a. Ведёт клиентскую базу, добавляет и редактирует карточки клиентов;
    b. Оформляет и изменяет заказы клиентов;

c. Контролирует статусы заказов и взаимодействует с бухгалтерией по вопросам оплаты;
d. Формирует коммерческие предложения и рассчитывает итоговую стоимость заказа.

3. Кладовщик:
    a. Учёт поступления товаров на склады и оформление приёмочных документов;
    b. Отслеживание остатков и проведение инвентаризаций;
    c. Перемещение товаров между складами;
    d. Подготовка отгрузок по заказам и оформление отгрузочных документов.
4. Бухгалтер:
    a. Формирование и отправка клиентам счетов-фактур;
    b. Учёт поступивших платежей и сверка по задолженностям;
    c. Ведение налогового учёта и расчёт налоговых обязательств;
    d. Подготовка финансовой отчётности и аналитических сводок.
5. Гость (неавторизованный пользователь):
    a. Просмотр каталога товаров, прайс-листов, публичной информации о компании;
    b. Ознакомление с условиями доставки и оплаты;
    c. Подача предварительных запросов на получение коммерческого предложения.

2.6. 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:
    a. Статусы сущностей (фильтрация заказов, счетов, поставок, заявок по статусам);
    b. Характеристики товаров (категория, артикул, наименование, диапазон цен, поставщик);
    c. Информация о контрагентах (название клиента или поставщика, ИНН, контактное лицо);
    d. Даты (дата заказа, дата поставки, дата выставления счета, срок оплаты, даты перемещений на складе);
    e. Сотрудники (ФИО, должность, подразделение) — для быстрого поиска исполнителей задач и ответственных лиц.

2.7. 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).

2.8. 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.

3. ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

3.1. 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

3.2. Особенности нормализации

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

1. Атомарность атрибутов
    a. Имя и фамилия сотрудников, клиентов и поставщиков представлены отдельными полями first_name и last_name для упрощения поиска, сортировки и локализации по фамилии.
    b. 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.
    c. 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
    a. Заказы (orders) и позиции заказов (order_items) хранятся в разных таблицах, так как один заказ может содержать множество товарных строк.
    b. Счета (invoices) и позиции счетов (invoice_items) также выделены в отдельные таблицы для точного отражения деталей расчетов и поддержки нескольких позиций на одном счете.
    c. Клиенты (customers) и поставщики (suppliers) хранятся в отдельных таблицах, несмотря на сходную структуру, чтобы разграничить процессы продажи и закупки.
    d. 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.
    e. Уведомления (notifications) и получатели (notification_recipients) вынесены в отдельные таблицы.

4. ПОДРОБНОЕ ОПИСАНИЕ БИЗНЕС-ЛОГИКИ

4.1. 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;
```

2. 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;
```

3. 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;
```

4. 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);
```

5. 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;
```

6. 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;
```

4.2. 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 ;
```

2. 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 ;
```

3. 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 ;
```

4. 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 ;
```

5. 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 ;
```

4.3. 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 ;
```

2. 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 ;
```

3. 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 ;
```

4. 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 ;
```

5. 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 ;
```

6. 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 ;
```

7. 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 ;
```

8. 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 ;
```

9. 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 ;
```

10. 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 ;
```

11.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 ;
```

12.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 ;
```

13.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 ;
```

5. ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

5.1. 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

5.2. 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

5.3. 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

5.4. 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);
```