**Дипломный проект по профессии Инженер данных**

**1.Цель проекта**.

Разработка системы извлечения, трансформации и загрузки данных (ETL) в хранилище данных с последующим построением аналитической витрины и визуализации ключевых метрик.

**2. Используемые технологии.**

**Язык программирования:** Python (pandas, sqlalchemy)

**СУБД:** PosgreSQL

**Оркестрация ETL**: Apache Airflow

**BI-инструмент:** Power BI

**IDE (набор ПО для создания кода):** Dbeaver, VSCode, Jupyter Notebook

**3. Структура хранилища данных.**

Основной источник данных — CSV-файл, содержащий информацию о продажах: транзакции, клиенты, товары, филиалы, даты и пр. Всего 1000 строк, структура таблицы включает такие поля, как:

invoice\_id, branch, city, customer\_type, gender, product\_line, unit\_price, quantity', tax\_5\_percent, total, date, time, payment\_method, cogs, gross\_margin\_percentage, gross\_income, rating.

**3. Описание слоёв хранилища.**

**3.1. Слой NDS (Normalized Data Store) нормализованное хранилище**

Созданы следующие нормализованные таблицы:

**customer** — информация о клиентах (тип и пол);

customer\_id-первичный ключ с автоинкриментом,

customer\_type – тип клиента,

gender-пол клиента.

**products** — товары и цены;

product\_id (PK) -первичный ключ с автоинкриментом,

product\_line – категории товаров,

unit\_price-стоимость товара за единицу.

**branches** — филиалы и города;

branch\_id- первичный ключ с автоинкриментом,

branch\_code-номер филиала,

city-город расположение филиала.

**dates** — дата и календарные показатели;

full\_date-полная дата,

day-день,

month-месяц,

year-год,

weekday-день недели.

**sales** — факты продаж со связями на остальные таблицы через внешние ключи;

invoice\_id- первичный ключ, номер счет-фактуры,

customer\_id-внешний ключ из таблицы customer,

products-внешний ключ из таблицы products,

branch\_id-внешний ключ из таблицы branches,

date\_id-внешний ключ из таблицы dates,

quantity-количество товаров,

tax\_5\_percent-налог 5%,

total-сумма покупки,

payment\_method-метод оплаты,

cogs-себестоимость,

gross\_margin\_percentage-процент маржи,

gross\_income-валовая прибыль,

rating-Оценка.

**3.2. Слой DDS (Dimensional Data Store) схема звезда.**

Слои построенные по звёздной схеме:

**dim\_customer** — измерение по клиенту;

customer\_id-первичный ключ с автоинкриментом,

customer\_type – тип клиента,

gender-пол клиента.

**dim\_product** — измерение по продукту;

product\_id (PK) -первичный ключ с автоинкриментом,

product\_line – категории товаров,

unit\_price-стоимость товара за единицу.

**dim\_branch** — измерение по филиалу;

branch\_id- первичный ключ с автоинкриментом,

branch\_code-номер филиала,

city-город расположение филиала.

**dim\_date** — измерение по дате;

full\_date-полная дата,

day-день,

month-месяц,

year-год,

weekday-день недели.

**fact\_sales** — таблица фактов продаж с расчётными полями:

invoice\_id- первичный ключ, номер счет-фактуры,

customer\_id-внешний ключ из таблицы dim\_customers,

products-внешний ключ из таблицы dim\_products,

branch\_id-внешний ключ из таблицы dim\_branch,

date\_id-внешний ключ из таблицы dim\_date,

quantity-количество товаров,

tax\_5\_percent-налог 5%,

total-сумма покупки,

payment\_method-метод оплаты,

cogs-себестоимость,

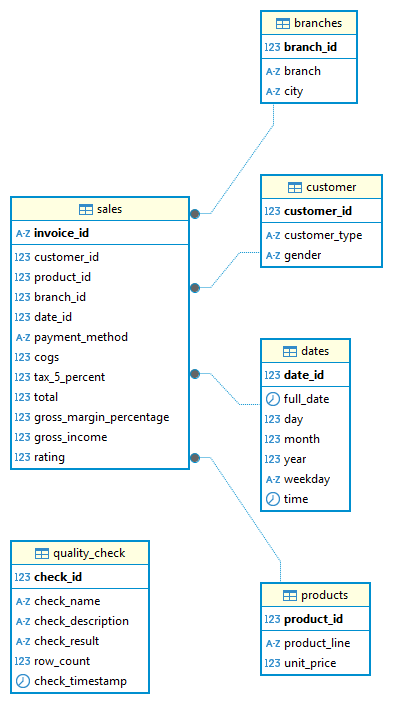
gross\_income-валовая прибыль,

rating-Оценка.

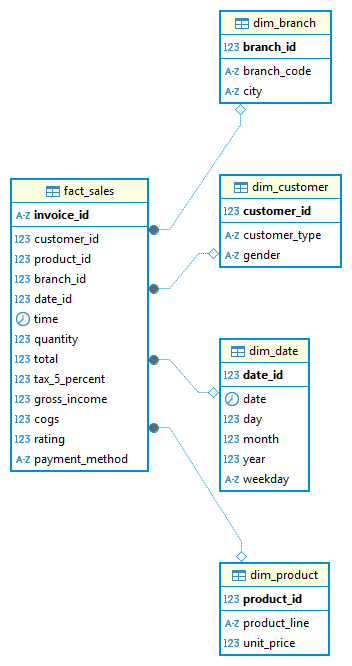
**4. ER-диаграммы.**

В качестве ER-диаграмм использованы схемы, сгенерированные в DBeaver. Они показывают связи между таблицами NDS и DDS и визуально иллюстрируют структуру хранилища.

ER-диаграмма схемы NDS.



ER-диаграмма схемы DDS.



**5. ETL-процессы.**

Реализованы с помощью Python (pandas + psycopg2) и SQL:

Загрузка и очистка данных из CSV;

Проверка и удаление дубликатов по invoice\_id;

Заполнение нормализованных таблиц (NDS);

Формирование витрин данных (DDS) с использованием surrogate keys и агрегаций.

**6. Оркестрация**

Оркестрация процессов реализована с использованием Apache Airflow. Настроены DAG`и, выполняющие:

load\_to\_normalized\_data – обработка и загрузка данных в нормализованную таблицу;

quality\_check - проверка качества данных;

load\_to\_dimension\_data - загрузка данных в витрины;

master\_dag – код для запуска dag`ов по очереди загрузку и трансформацию данных.

**7. Качество данных Data quality.**

Проведены проверки на Python-скрипте, показаны выводы в отдельно созданной таблице по проверке качества данных и интегрирован DAG для оркестрации:

Проверка пропущенных значений;

Проверка дубликатов по invoice\_id;

Проверка на наличие отрицательных значений.

**8. Визуализация в Power BI.**

Созданы дашборды на основе данных из DDS в Power BI:

# Сумма и процент продаж по городам

# Общая сумма продаж по месяцам

# Сумма продаж по категориям товаров и по типу клиента

# Процент продаж по методу оплаты и по типу покупателя

**9.Особенности реализации.**

Все таблицы создаются программно через SQLAlchemy.

DAG полностью автоматизирует цепочку от загрузки до DDS.

Уникальные surrogate keys с автоинкрементом.

Структура данных документирована с помощью ER-диаграмм (DBeaver).

**10. Выводы.**

Реализована полноценная ETL-система с оркестрацией.

Данные хранятся в PostgreSQL, витрина построена в DDS.

Построены BI-дашборды в Power BI на основе DDS.

Подготовлена финальная документация для защиты диплома.

Дженишбеков Нурболот

DEG-37