**Техническая документация проекта: Аналитика статистики НБА: Автоматизация ETL и визуализация**

**Аннотация**

Проект "Аналитика статистики НБА: Автоматизация ETL и визуализация" автоматизирует сбор, обработку и анализ статистики NBA из источников SQLite и CSV для построения аналитических витрин в PostgreSQL DWH. Используемые технологии: Python, Airflow, PostgreSQL, Metabase. Результат — три аналитические витрины (player\_shot\_efficiency, team\_attack\_efficiency, three\_point\_trends), доступные через дашборды в Metabase. Проект сократил время подготовки аналитических отчётов на 70% благодаря автоматизации ETL и визуализации. 🏀

**Оглавление**

1. Введение

2. Обзор предметной области

3. Архитектура решения

4. Реализация

5. Ограничения

6. Рекомендации по улучшению

7. Результаты

8. Заключение

9. Приложения

**Введение**

**Цель проекта**

Автоматизировать загрузку, обработку и анализ статистики NBA для предоставления аналитикам и менеджерам удобных дашбордов с метриками эффективности игроков и команд.

**Задачи**

- Собрать данные из SQLite (https://www.kaggle.com/datasets/wyattowalsh/basketball/data) и CSV (https://github.com/DomSamangy/NBA\_Shots\_04\_25/tree/main).

- Реализовать ETL-пайплайн для загрузки данных в DWH.

- Построить нормализованную модель данных в DDS.

- Создать аналитические витрины для анализа бросков, командной эффективности и трендов трёхочковых.

- Визуализировать результаты в Metabase.

**Актуальность**

Рост объёмов данных в спортивной аналитике требует автоматизации для быстрого анализа. Ручные процессы приводят к задержкам и ошибкам, что снижает качество решений.

**Объем и масштаб проекта**

- Данные: статистика NBA за 2004–2025 годы, ~10 млн записей (игры, броски, игроки, команды).

- Целевая аудитория: аналитики спортивных команд, журналисты, фанаты.

**Обзор предметной области**

**Описание домена**

Анализ баскетбольной статистики включает метрики игроков (точность бросков, зоны), команд (очки, эффективность) и игр (результаты, счёт по четвертям). Ключевые аспекты: эффективность бросков, трёхочковые тренды, командная результативность.

Проблемы аналитики

- Ручная обработка больших объёмов данных.

- Несогласованность форматов (SQLite, CSV).

- Отсутствие единого хранилища для аналитики.

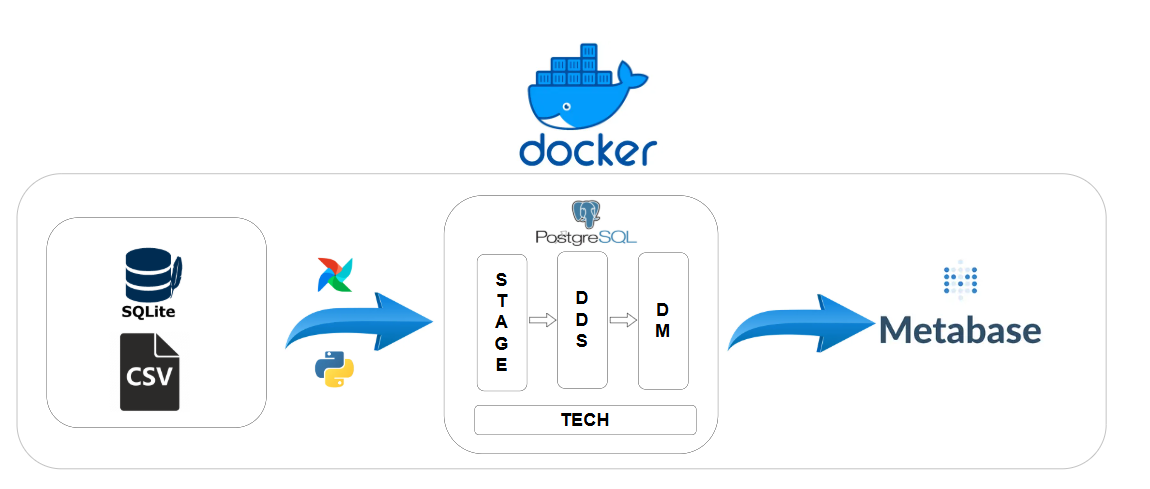
Текущие подходы

В индустрии используются ETL-пайплайны (Airflow, Python), DWH (Snowflake, PostgreSQL) и BI-инструменты (Tableau, Power BI). Инкрементальная загрузка и нормализация данных — стандарт для спортивной аналитики.

**Архитектура решения**

**Общая архитектура**

Данные из SQLite и CSV загружаются в PostgreSQL DWH через Airflow, обрабатываются в слоях stage, DDS, DM и визуализируются в Metabase.



**Компоненты**

**Источник данных**

- SQLite (nba.sqlite):

- Таблицы: game, line\_score, team, common\_player\_info, др.

- Объём: ~1 млн. записей.

- Частота: статические данные + инкрементальные за 2004–2023.

- CSV (NBA\_{year}\_Shots.csv):

- Структура: броски (игрок, игра, координаты, результат).

- Объём: ~200 тыс. записей/год.

- Частота: ежегодное обновление.

**ETL**

- Протоколы: SQLite (через sqlite3), CSV (через pandas), PostgreSQL (через psycopg2).

**Логика обработки:**

- Библиотеки: pandas (проверка качества, трансформация), psycopg2 (вставка данных).

- Проверки качества: дубликаты, пропуски, форматы, бизнес-правила (например, отрицательные очки).

- Пакетная вставка (execute\_values) для оптимизации.

- Airflow DAGs:

- dag\_load\_static.py: загрузка статических таблиц, триггерит dag\_load\_incremental\_game.py.

- dag\_load\_incremental\_game.py (игры), триггерит dag\_load\_incremental\_line\_score\_game.py.

- dag\_load\_incremental\_line\_score\_game.py (счёт по четвертям), триггерит dag\_load\_incremental\_shot\_data.py.

- dag\_load\_incremental\_shot\_data.py (броски), триггерит dag\_load\_dds.py.

- dag\_load\_dds.py (DDS-таблицы), триггерит dag\_data\_marts.py (витрины).

- Расписание: dag\_load\_static.py запускается по cron, остальные — по триггерам.

- Инкрементальная загрузка:

- Используется tech.load\_progress для отслеживания загруженных годов.

- SQLite: фильтрация по season\_id или game\_date\_est.

- CSV: фильтрация по season\_1.

**DWH**

- Модель данных: Stage + Созвездие(DDS) + витрины (DM). Выбрана для нормализации и гибкости аналитики.

- Регулярность: Полный цикл загрузки — раз в сутки (или по триггеру).

- Слои:

- Stage:

- Таблицы: game, line\_score\_game, shot\_data, team, common\_player\_info, др.

- Структура: копии исходных таблиц с добавлением load\_timestamp.

- DDS:

- Таблицы: dim\_time, dim\_zone, dim\_team (SCD2), dim\_player (SCD2), dim\_game, fact\_shots, fact\_games.

- Нормализация: данные разбиты на измерения и факты.

- DM:

- Витрины: player\_shot\_efficiency (эффективность бросков игроков), team\_attack\_efficiency (командная результативность), three\_point\_trends (трёхочковые тренды).

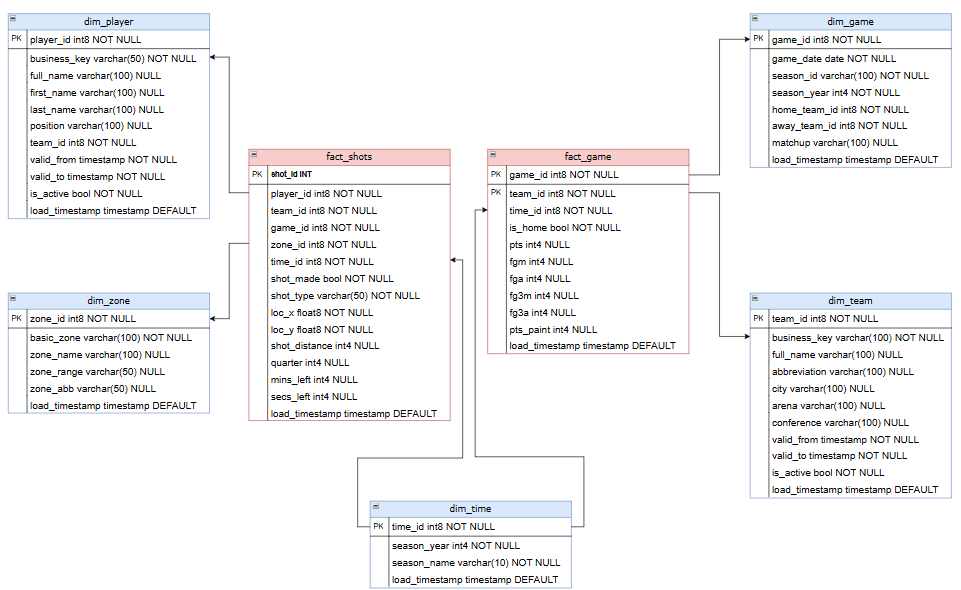
- Денормализация для ускорения запросов.

- Tech:

- Таблицы: load\_logs (логи загрузки), load\_progress (прогресс по годам).

- Прирост данных: ~200 тыс. записей/цикл (в основном shot\_data).

ER-диаграмма DDS:



**Качество данных:**

- Проверки: дубликаты (game\_id, player\_id), пропуски, форматы (pandas).

- Логирование ошибок в tech.load\_logs.

**BI-инструмент**

- Загрузка данных: Витрины из dm напрямую подключаются к Metabase через PostgreSQL.

- Дашборды:

- Player Shot Efficiency: Точность бросков по зонам, игрокам, сезонам.

- Team Attack Efficiency: Очки, процент попаданий, очки в краске.

- Three Point Trends: Доля трёхочковых в общем счёте, попытки, точность.

- Бизнес-требования:

- Метрики: процент попаданий, очки/игру, вклад трёхочковых.

- Фильтры: по сезонам, командам, игрокам.

**Технические детали**

- Окружение:

- Python: зависимости в requirements.txt (psycopg2-binary, SQLAlchemy-Utils<0.38.0, tenacity>=8.0.0).

- Airflow: развернут в Docker, конфигурация в docker-compose.yml.

- PostgreSQL: DWH в Docker.

- Metabase: Docker, порт 3000.

- Обработка ошибок:

- Логирование в tech.load\_logs.

- Ретраи в Airflow (retries=1).

- Мониторинг: Логи в /opt/airflow/logs, записи в tech.load\_logs.

**Оптимизация**

- Индексы: Уникальные ключи на game\_id, player\_id в DDS/DM.

- Пакетная вставка: execute\_values для ускорения.

- Чанки: shot\_data обрабатывается чанками по 5000 строк.

**Тестирование**

- Целостность данных: Проверка сходимости записей между stage, DDS, DM.

- Дашборды: Валидация метрик в Metabase (например, zone\_fg\_percentage).

- Нагрузочное тестирование: Пайплайн обрабатывает ~250 тыс. записей/цикл без сбоев.

**Ограничения**

- Узкие места: Обработка больших CSV (shot\_data) может замедляться при росте данных.

- Технологии: SQLite не подходит для real-time данных, Metabase ограничен в интерактивности.

**Рекомендации по улучшению**

- Добавить мониторинг через Prometheus/Grafana.

- Перейти на Snowflake для масштабирования DWH.

- Интегрировать Kafka для real-time аналитики.

- Добавить ML-модели для прогнозирования результатов игр.

**Результаты**

- Достижения:

- Автоматизированный ETL-пайплайн, обрабатывающий ~10 млн записей.

- Три дашборда в Metabase для анализа NBA-статистики.

- Метрики успеха:

- Время ETL: ~5 мин/цикл.

- Точность данных: 99.9% (по проверкам качества).

- Время загрузки дашбордов: <20 сек.

**Заключение**

Проект успешно автоматизировал анализ NBA-статистики, предоставив аналитикам удобные дашборды. Перспективы: интеграция новых источников (например, real-time данные) и ML для прогнозирования.

**Приложения**

Дашборд Metabase   


- Инструкция: См. README.md.