# Лунапаркова Лудница Документация

## Overview

Документа съдържа информация за структура на база от данни за управление на лунапарк и обяснява принципа на работа на Python скрипт, който е създаден за генериране на данни за базата данни.

Структурата на базата от данни предоставя възможност за:

* Продажба на билети;
* Проследяване на статус на Атракциони;
* Категоризиране на различни заведения (магазини, ресторанти);
* Организация на служители;
* Информация за всички продажби от различни заведения;
* Информация за всички транзакции (продажни, рекламации);
* Организация на промоции

## Схема на Базата от данни и пояснения към всяка таблица

A screenshot of a computer

AI-generated content may be incorrect.

**Фигура 1 ER Диаграма на Базата от данни**

1. **VISITORS:**

Съхранява информация за всеки посетител (име, имейл, телефон…);

1. **TICKETS:**

Съхранява информация за всеки продаден билет (тип на билета, кой потребител го е купил)

1. **ATTRACTIONS**:   
   Съхранява информация за различните атракции – тип, текущ статус, капацитет и дата на създаване.
2. **ROLES**:   
   Описва роли на служителите, като мениджър, касиер и т.н.
3. **EMPLOYEES**:   
   Служители на организацията с техните имена, контакти, роли, дата на наемане, заплата и статус.
4. **ESTABLISHMENTS**:   
   Данни за заведения или магазини – име, категория, дата на създаване.
5. **PRODUCTS**:   
   Продукти, които се продават в заведенията и магазините – име, категория, цена и връзка със заведението, в което се предлагат.
6. **PROMOTIONS**:   
   Активни промоции, процент на отстъпка и период на валидност.
7. **SALES**:   
   Продажби на продукти, количество, обща цена, дата на продажба, свързани с продукт и евентуално промоция.
8. **TRANSACTIONS**:   
   Финансови транзакции, включително тип (покупка на билет, продукт или възстановяване на пари), дата и сума, както и връзки с посетителите, служителите и продажбите.

## Временни (transient) таблици

Временните таблици са аналогични на реалните без дефинирани връзки между тях. Тяхната роля е междинно да съхраняват информацията и да валидират записите, тези които не отговарят на изискванията остават там, за да бъдат анализирани и при нужда да се трансформират да отговарят на изискванията.  
  
Този подход е полезен, защото:

* Лесно могат да се проследят и коригират невалидни записи;
* Подобрява се качеството на данните, които се вкарват в базата от данни

## Работа на Python script

Процесът включва следните стъпки:

1. **Избор на бизнес климат**: Потребителят избира позитивен или негативен бизнес климат, който влияе върху броя на генерираните записи.
2. **Генериране на данни**: Python скрипта автоматично създава фалшиви записи за различни обекти и ги записва във файлове в CSV формат.
3. **Импортиране на данни в Snowflake**: CSV файловете се качват в Snowflake.
4. **Записване във временни (transient) таблици**: Данните се прехвърлят от CSV файловете в временни таблици, където подлежат на първоначално филтриране и проверка.
5. **Прехвърляне на валидни данни**: Само валидните записи, които отговарят на зададените бизнес условия и правила, се прехвърлят от временните таблици в окончателните (реални) таблици.
6. **Почистване на временните таблици**: Валидните записи се премахват от временните таблици, като остават само невалидните, които могат да се обработят допълнително за постигане на валидност.
7. **Почистване на пространството**: CSV файловете се изтриват от Snowflake за оптимизация на пространството.