

|  |
| --- |
| Subway Business Template  **Subject areas** |
| **Logo / Image** |

Contents

[1 Business Description 3](#_Toc62212630)

[1.1 Business Background 3](#_Toc62212631)

[1.2 Problems. Current Situation 3](#_Toc62212632)

[1.3 The benefits of implementing a database. Project Vision 3](#_Toc62212633)

[2 Model description 3](#_Toc62212634)

[2.1 Definitions & Acronyms 3](#_Toc62212635)

[2.2 Logical Scheme 3](#_Toc62212636)

[2.3 Objects 3](#_Toc62212637)

# 

# Business Description

## Business background

Efficient management of a metro station system is crucial for urban transport networks. Metro stations require precise coordination across various operational dimensions, including train schedules, ticket sales, employee management, and infrastructure maintenance such as tunnels, tracks, and stations. Effective management ensures safety, reliability, and optimal passenger experience, contributing to urban mobility's overall efficiency and sustainability.

## Problems. Current Situation

Metro management faces several challenges due to manual data handling and disconnected systems. Key issues include:

* Inefficient scheduling causing delays and overcrowding.
* Inaccurate tracking of maintenance activities leading to infrastructure failures.
* Limited capability to analyze ticket sales and promotions effectively.
* Poor employee assignment processes, causing resource misallocation.

These issues increase operational costs, passenger satisfaction, and enlarge safety risks.

## the Benefits of implementing a database. Project Vision

Implementing a database system will:

1. Improve schedule accuracy and operational efficiency.
2. Enhance infrastructure maintenance tracking, reducing unexpected downtime.
3. Provide insights into passenger preferences and promotional effectiveness.
4. Facilitate better resource allocation through improved employee management.

# Model description

## Definitions & Acronyms

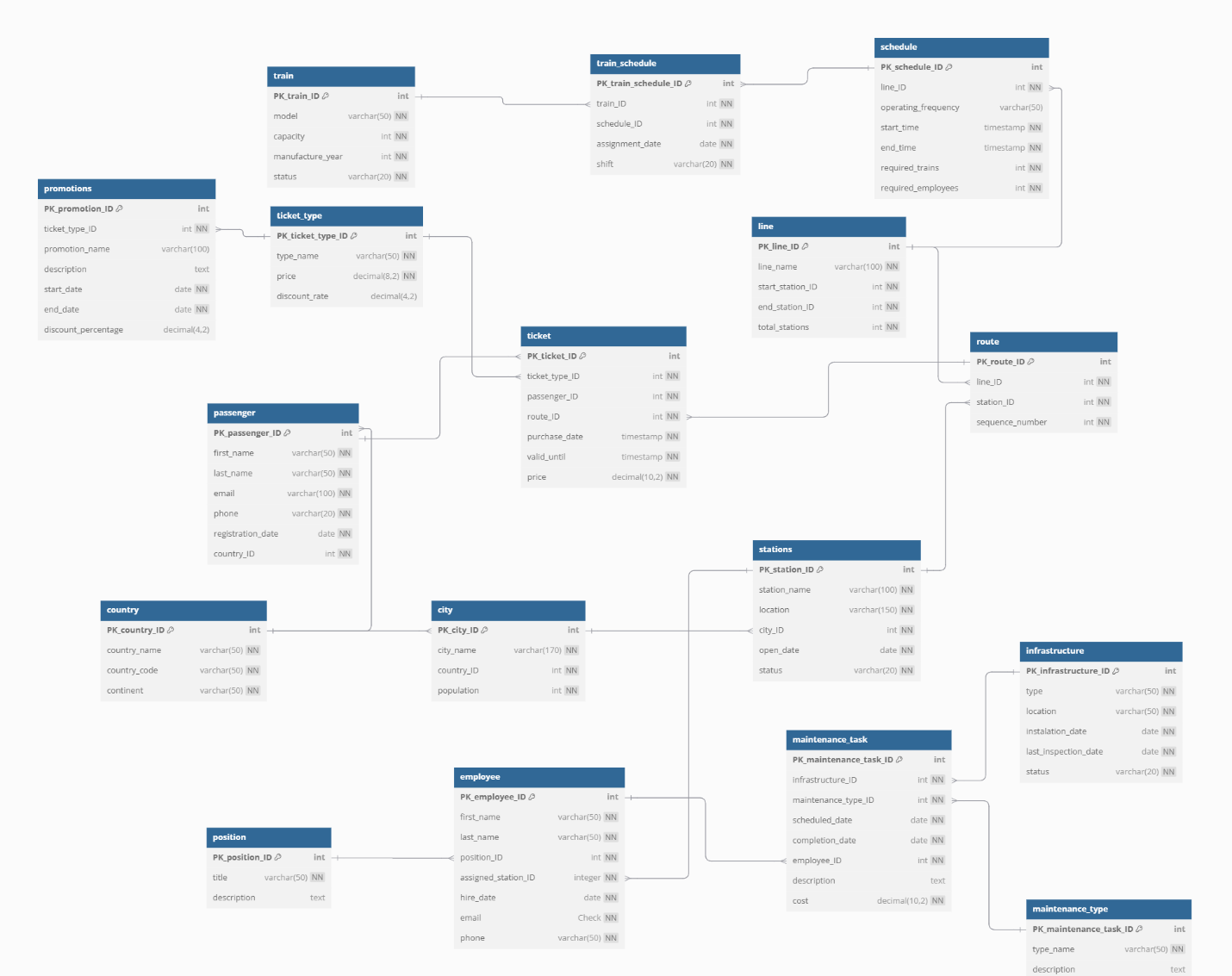
**PK**: Primary Key

**FK**: Foreign Key

**Operational Frequency**: How often trains run on a specific line.

**Infrastructure**: Physical assets of the metro such as tunnels, tracks, and stations.

## Logical Scheme



## Objects

Table Description

1) This table stores data about stations.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| stations | station\_ID | PK, auto-increment | Int |
| name  location  open\_date  status | Not Null  Not Null  Not Null  Not Null | Varchar(100)  Varchar(150)  Date  Varchar(20) |

Comments on table relationships:

City (many-to-one): Each station is located in one specific city.

Line (one-to-many): Stations serve as starting or ending points for multiple lines.

Route (one-to-many): Stations are included in multiple routes.

Employee (one-to-many): Stations have multiple employees assigned to them.

Example with data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| station\_ID | name | location | open\_date | status |
| 1 | Didube | Tbilisi | 01/10/1961 | Functional |

2)This table describes each metro line, including starting and ending stations

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| line | line\_ID | PK, auto-increment | Int |
| name  start\_station\_ID  end\_station\_ID  total\_stations | Not Null  FK, Not Null  FK, Not Null  Not Null | Varchar(100)  Int  Int  Int |

Comments on table relationships:

Station (many-to-one): Each line has exactly one start and one end station.

Route (one-to-many): A line comprises multiple routes.

Schedule (one-to-many): A line can have multiple operational schedules

Example with data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| line\_ID | name | start\_station\_ID | end\_station\_ID | total\_stations |
| 1 | Saburtalo Line | 4 | 36 | 7 |

3) This table defines the sequence and association of stations within specific metro lines.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| route | route\_ID | PK, auto-increment | Int |
| line\_ID  station\_ID  sequence\_number | FK, Not Null  FK, Not Null  Not Null | Int  Int  Int |

Comments on table relationships:

ine (many-to-one): Each route belongs exclusively to one metro line.

Station (many-to-one): Each route step refers to exactly one station.

Ticket (one-to-many): Passengers purchase tickets specific to a route.

Example with data

|  |  |  |  |
| --- | --- | --- | --- |
| route\_ID | line\_ID | station\_ID | sequence\_number |
| 1 | 1 | 5 | 12 |

4) This table contains information about individual trains including capacity and operational status.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| train | train\_ID | PK, auto-increment | Int |
| model  capacity  manufacture\_year  status | Not Null  Not Null  Not Null  Not Null | Varchar(50)  Int  Year  Varchar(20) |

Comments on table relationship:

**TrainSchedule (one-to-many)**: Trains are assigned to multiple schedules.

Example with data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| train\_ID | model | capacity | manufacture\_year | status |
| 1 | MAN | 200 | 2004 | Functional |

5) This table maintains data regarding employees, their positions, and assigned stations.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| employee | employee\_ID | PK, auto-increment | Int |
| first\_name  last\_name  position\_ID  assigned\_station\_ID  hire\_date  email  phone | Not Null  Not Null  FK, Not Null  FK, Not Null  Not Null  Unique, NOT NULL, CHECK (email LIKE '%*@*%.\_%')  Not Null, Unique | Varchar(50)  Varchar(50)  Int  Int  Date  Varchar(100)  Varchar(20) |

Comments on table relationships:

Position (many-to-one): Employees hold exactly one position.

Station (many-to-one): Employees are assigned to exactly one station.

MaintenanceTask (one-to-many): An employee handles multiple maintenance tasks.

Example with data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| employee\_ID | first\_name | last\_name | position\_ID | assigned\_station\_ID | hire\_date | email | phone |
| 1 | Giorgi | Mujrii | 4 | 2 | 3/15/2021 | skop@gmail.com | +995123123 |

6) This table lists all employee positions and their detailed descriptions.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| position | position\_ID | PK, auto-increment | Int |
| title  description | Not Null  Not Null | Varchar(50)  Text |

Comments on table relationship:

Employee (one-to-many): Multiple employees can occupy the same position.

Example with data:

|  |  |  |
| --- | --- | --- |
| position\_ID | title | description |
| 1 | Train Operator | Responsible for operating metro trains. |

7) This table holds schedules for each metro line including operational timings and resource requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| schedule | schedule\_ID | PK, auto-increment | Int |
| line\_ID  operating\_frequency  start\_time  end\_time  required\_trains  required\_employees | Not Null  Not Null  Not Null  Not Null  Not Null  Not null | Int  Varchar(50)  Timestamp  Timestamp  Int  Int |

Comments on table relationships:

Line (many-to-one): Each schedule corresponds to one specific line.

TrainSchedule (one-to-many): Schedules have multiple trains assigned through train schedules.

Example with data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| schedule\_ID | line\_ID | operating\_frequency | start\_time | end\_time | required\_trains | required\_employees |
| 1 | 4 | Everyday | 3/15/2025  07:00 Am | 3/15/2025  11:00 PM | 4 | 12 |

8) This table is junction table. It was created because train and schedule had many-to-many relationship. It records the assignment of trains to specific schedules and shifts.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| train\_schedule | train\_schedule\_ID | PK, auto-increment | Int |
| train\_ID  schedule\_ID  assignment\_date  shift | Fk, Not Null  FK, Not Null  Not Null  Not Null | Int  Int  Date  Varchar(20) |

Comments on table relationships

Train (many-to-one): Each train schedule references exactly one train.

Schedule (many-to-one): Each train schedule entry references exactly one schedule.

Example with data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| train\_schedule\_ID | train\_ID | schedule\_ID | assignment\_date | shift |
| 1 | 4 | 12 | 3/14/2025 | Morning (6:00 – 14:00) |

9) This table tracks ticket transactions and links passengers to specific routes.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| ticket | ticket\_ID | PK, auto-increment | Int |
| ticket\_type\_ID  passenger\_ID  route\_ID  purchase\_date  valid Until  price | Fk, Not Null  FK, Not Null  FK, Not Null  Not Null  Not Null  Not null | Int  Int  Int  Timestamp  Timestamp  Decimal(10,2) |

Comments on table relationships:

TicketType (many-to-one): Each ticket is categorized under exactly one ticket type.

Passenger (many-to-one): Each ticket belongs exactly to one passenger.

Route (many-to-one): Each ticket is issued specifically for one route.

Example with data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ticket\_ID | icket\_type\_ID | passenger\_ID | route\_ID | purchase\_date | valid\_until | price |
| 13 | 3 | 104 | 13 | 3/15/2025  11:00 PM | 3/16/2025  11:00 AM | 12,49 |

10) This table includes the details of various ticket types, their base prices, and discounts available.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| ticket\_type | ticket\_type\_ID | PK, auto-increment | Int |
| type\_name  price  discount\_rate | Not Null  Not Null  Not Null | Varchar(50)  Decimal (8,2)  Decimal (4,2) |

Comments on table relationships:

Ticket (one-to-many): A ticket type can apply to multiple tickets.

Promotion (one-to-many): Multiple promotions can apply to one ticket type

Example with data:

|  |  |  |  |
| --- | --- | --- | --- |
| ticket\_type\_ID | type\_name | price | discount\_rate |
| 1 | Regular | 49,99 | 10,5 |

11) This table manages promotional campaigns associated with different ticket types.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| promotions | promotion\_ID | PK, auto-incerement | Int |
| ticket\_type\_ID  promotion\_name  description  start\_date  end\_date  discount\_percentage | FK, Not Null  Not Null  Not Null  Not Null  Not null | Int  Varchar(100)  Text  Date  Date  Decimal(4,2) |

Comments on table relationship:

TicketType (many-to-one): Each promotion is linked exclusively to one ticket type.

Example with data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| promotion\_ID | ticket\_type\_ID | promotion\_name | description | start\_date | end\_date | discount\_percentage |
| 1 | 1 | Christmas Promotion | Promotion for Christmas holidays | 12/15/2024 | 01/01/2025 | 20,00 |

12) This table records scheduled and completed maintenance tasks for metro infrastructure.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| maintenance\_task | maintenance\_task\_ID | PK, auto-incerement | Int |
| infrastructure\_ID  maintenance\_type \_ID  scheduled\_date  completion\_date  employee\_ID  description  cost | FK, Not Null  FK, Not Null  Not Null  Not Null  FK, Not Null  Not Null | Int  Int  Date  Date  ID  Text  Decimal(10,2) |

Comments on table relationships:

Infrastructure (many-to-one): Each task targets exactly one infrastructure component.

MaintenanceType (many-to-one): Each task has one specific maintenance type.

Employee (many-to-one): Each task is assigned to exactly one employee.

Example with data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| maintenance\_task\_ID | infrastructure\_ID | maintenance\_type\_ID | scheduled\_date | completion\_date | employee\_ID | description | cost |
| 1 | 15 | 3 | 12/15/2024 | 01/15/2025 | 01/01/2025 | 20,00 | 15000,00 |

13) This table categorizes types of maintenance tasks with relevant descriptions.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| maintenance\_type | maintenance\_type\_ID | PK, auto-increment | Int |
| type\_name  description | Not Null | Varchar(50)  Text |

Comments on table relationships:

MaintenanceTask (one-to-many): A maintenance type can have multiple tasks categorized under it.

Example with data:

|  |  |  |  |
| --- | --- | --- | --- |
| maintenance\_type\_ID | type\_name | description | |
| 1 | Escalator Maintenance | | Changing the damaged escalator fully |

14) This table details infrastructure components such as tunnels, tracks, and stations, along with their condition and maintenance history.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Infrastructure | infrastructure\_ID | PK, auto-increment | Int |
| type  location  installation\_date  inspection\_date  status | Not Null  Not Null  Not Null  Not Null  Not Null | Varchar (50)  Varchar (150)  Date  Date  Varchar (20) |

Comments on table relationship:

**MaintenanceTask (one-to-many)**: Infrastructure components have multiple maintenance tasks associated with them.

Example with data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| infrastructure\_ID | type | location | instalation\_date | inspection\_date | status |
| 12 | Escalator | At Metro Station Didube | 01/01/2014 | 01/15/2025 | Functional |

15) This table stores passenger information including contact details and registration dates.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Passenger | passenger\_ID | PK, auto-incerement | Int |
| first\_name  last\_name  email  phone  registration\_date | Not Null  Not Null  Unique, NOT NULL, CHECK (email LIKE '%*@*%.\_%')  Unique, Not null  Not Null | Varchar(50)  Varchar(50)  Varchar(100)  Varchar(20)  Date |

Comments on table relationships:

Country (many-to-one) (optional): Passengers originate from exactly one country.

Ticket (one-to-many): Passengers can buy multiple tickets.

Example with data:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| employee\_ID | First\_name | Last\_name | Email | Phone | Registration\_date |
| 17 | Anna | Kordzaia | skopio@gmail.com | +995123124 | 3/15/2021 |

16) This table stores information about countries.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Country | country\_ID | PK, auto-increment | Int |
| country\_name  country\_code  continent | Not Null  Not null  Not null | Varchar(50)  Varchar(3)  Varchar(50) |

Comments on table relationships:

City (one-to-many): A country contains multiple cities.

Passenger (one-to-many) (optional): Passengers originate from a specific country.

Example with data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| country\_ID | country\_name | country\_code | | continent |
| 1 | Georgia | | GEO | Europe |

17) This table stores information about cities.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| City | city\_ID | PK, auto-increment | Int |
| city\_name  country\_ID  population | Not Null  FK, Not null  Not null | Varchar(50)  Varchar(50)  Int |

Comments on table relationships:

Country (many-to-one): Each city belongs to exactly one country.

Station (one-to-many): A city hosts multiple metro stations.

Example with data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| city\_ID | city\_name | country\_ID | | population |
| 1 | Tbilisi | | 1 | 1300000 |