POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

First EICTA Project

Author(s): **Bamhaoud Younes - 10767795**
Calcara Antonio - 11095781
Foini Lorenzo - 10828129
Martinelli Francesco - 10767793

Group Number: **7**

Academic Year: 2024-2025

Contents

Contents **i**

1 Introduction **1**
1.1 Text delivery 1

2 Text Analysis: ER Model **3**

3 Logical Model **7**

4 Relational Model **13**

5 SQL requests **15**
5.1 Tables' definition 15
5.2 Database population 24
5.2.1 ChatGPT and Python 24
5.2.2 Data Screenshots 24
5.3 Queries 30

6 Conclusion **41**

1 | Introduction

The first project of the Enterprise ICT Architectures course concerns the definition of the ER, logical and relational models of a provided text and the subsequent definition of the database and querying it.

1.1. Text delivery

We now report the text to be analysed.
"The WellDressed fashion company wants to create a database about its products and exhibitions. The company hires fashion models to partake in fashion exhibitions. Each model provides their personal data, including Personal ID, name, surname, birth date, phone number, height, and dress size (XS, S, M, L, XL, etc.). Models are assigned to a fashion designer. These are described by their Personal ID, name, surname, stage name, phone number, and a brief description of their career. Fashion designers produce dresses for the models to wear in exhibitions. Each dress is worn by one model. Dresses are identified by a unique ID, size (XS, S, M, L, XL, etc.), colour, and production time. Dresses are made using fabrics described by a unique ID, price per square meter, name, and description of their properties. The amount of fabric used to produce a dress is also stored. Fabrics can be made from other fabrics. Models partake in exhibitions, described by a unique ID, duration (in minutes), title, location address, and description. Once a year, a fashion competition is held. The competition involves fashion designers submitting one of their dresses. The winning dress is marked as such. Dresses can be submitted to one competition only."

2 | Text Analysis: ER Model

After carefully analyzing the text, we identified the following entities and respective attributes:

- **fashion_designers:** personal_id, name, surname, phone_number, stage_name, career.
- **fashion_models:** personal_id, name, surname, phone_number, height, size, date_of_birth.
- **fashion_exhibitions:** id, duration, location_address, title, description.
- **fashion_competitions:** id, duration, location_address, title, description, year.
- **dresses:** id, size, colour, production_time.
- **fabrics:** id, price_per_square_meter, name, property (multiple attribute with cardinality 1:N).

These entities are connected to each other through the following relationships:

- **assigned_to:**
  - This relationship connects the entities fashion_designers and fashion_models.
  - Cardinality:
    - (1:N) from the fashion_designers side.
    - (1:1) from the fashion_models side.
- **wear:**
  - This relationship connects the entities fashion_models and dresses.
  - Cardinality:
    - (1:N) from the fashion_models side.
    - (1:1) from the dresses side.

- **partake_to:**
  - This relationship connects the entities fashion_models and fashion_exhibitions.
  - Cardinality:
    - (1:N) from the fashion_models side.
    - (1:N) from the fashion_exhibitions side.
- **shown_in:**
  - This relationship connects the entities dresses and fashion_competitions.
  - Cardinality:
    - (0:1) from the dresses side.
    - (1:N) from the fashion_competitions side.
  - Attribute:
    - winner: represents the victory or non-victory of the dress in the fashion competition.
- **composed_of:**
  - This relationship connects the entities dresses and fabrics.
  - Cardinality:
    - (1:N) from the dresses side.
    - (0:N) from the fabrics side.
  - Attribute:
    - amount: percentage of the fabric used to realise the dress.
- **made_of:**
  - This relationship connects the entities fabrics with itself.
  - Cardinality:
    - (0:N) from the "fabrics that are made of other fabrics" side.
    - (0:N) from the "fabrics that are used to make other fabrics" side.
  - Attribute:
    - amount: percentage of the fabric used to realise another fabric.

We also defined two ISA hierarchies, noting that:

- fashion_designers and fashion_models have four attributes in common, therefore grouped in the parent entity "people".

  → The hierarchy introduced is Total, Exclusive (T,E).

- fashion_competitions are a particular type of fashion_exhibitions and take place once a year.

  → The hierarchy introduced is Partial, Exclusive (P,E).

To define these entities, relationships and hierarchy, the following assumptions were used (therefore facts not present in the text, but which it makes sense for us to define for the correct implementation of the ER model):

- A model can only be hired by one designer and a designer must have at least one model.
- One model can wear multiple dresses.
- A dress is produced by exactly one designer and all dresses are given to models.
- The hierarchy between person and the two sub-classes is total (all registered people are either models or designers) and exclusive (no model can also be a designer and vice versa).
- Fashion competitions are a particular type of fashion exhibition.
- The properties of a fabrics do not have much importance in the model, so we represented them as multiple attribute and not as an entity.

In conclusion, Figure 2.1 represents the final version of the implemented ER model.

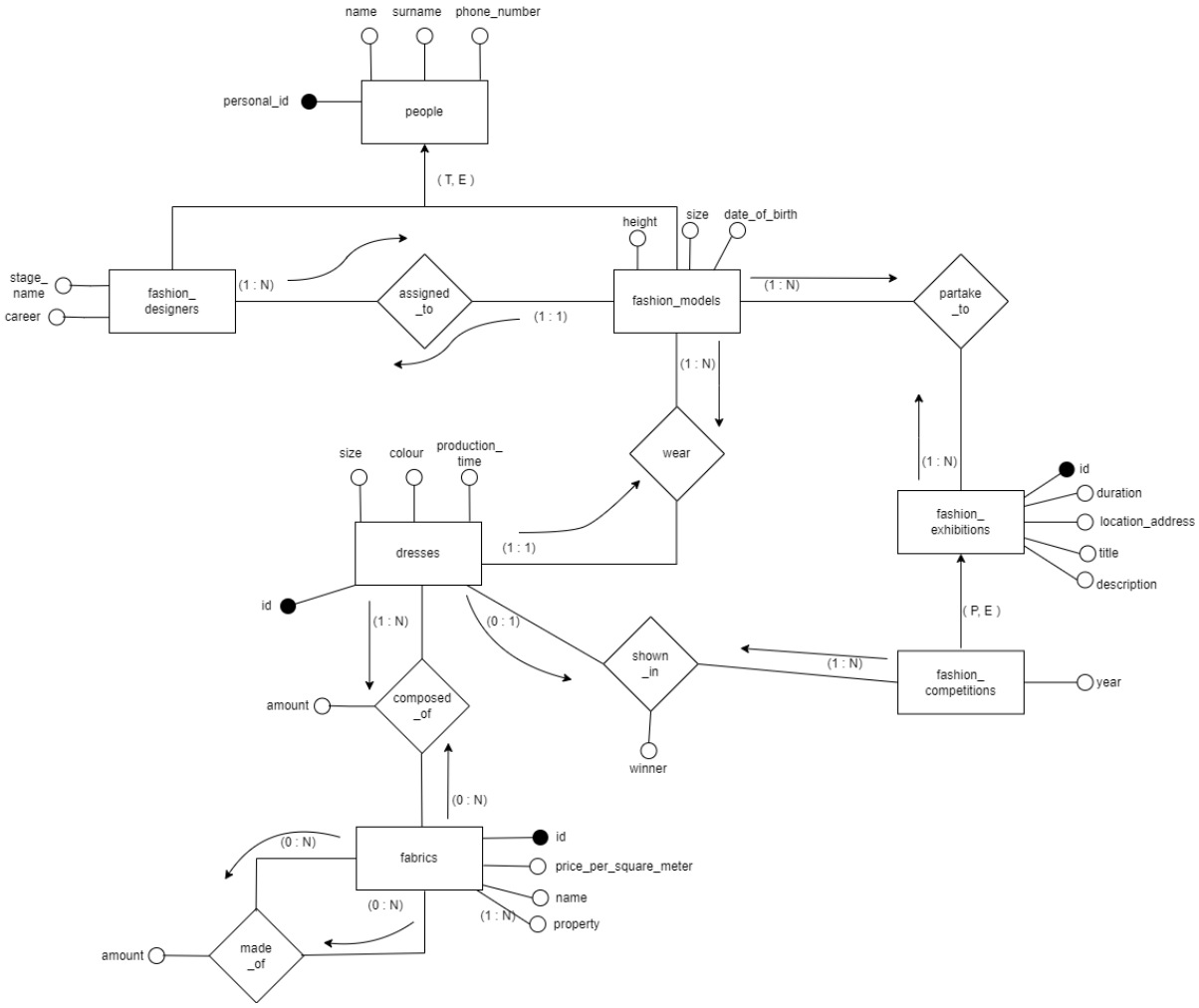

Figure 2.1: ER model.

3 | Logical Model

After having defined the ER model, we now proceed with the definition of the logical model, which consists of a logical translation of the ER model. Let's now analyse how all the entities defined in the ER model are transformed:

- **people:**
  Abstract entity that represents the general concept of person, it is specialized by the fashion_designers and fashion_models classes, analyzed later.
  It has the following attributes:
  - personal_id: unique integer identifier of the person (primary key).
  - name: string containing the person's name.
  - surname: string containing the person's surname.
  - phone_number: string containing the person's phone number.

  Since the hierarchy is (Total, Exclusive), the parent entity (i.e. people), is not translated into a table.

- **fashion_designers:**
  Entity representing fashion designers who propose fashion models for fashion exhibitions and design dresses to be shown in fashion competitions. This entity is a child of the "people" entity and consequently inherits all its attributes.
  Compared to the parent entity, it adds the following attributes:
  - stage_name: string containing the designer's stage name.
  - career: text description of the designer's career.

  Primary key: personal_id (inherited).

- **fashion_models:**
  Entity representing models who participate to fashion exhibitions. This entity is a child of the "people" entity and consequently inherits all its attributes.

Compared to the parent entity, it adds the following attributes:

- height: integer value in centimetres representing the model's height.
- size: string containing the model's size.
- date_of_birth: model's date of birth with the format "YYYY-MM-DD".
- fashion_designers_personal_id: reference to the personal id of the designer to whom the model is assigned.

Primary key: personal_id (inherited).
Foreign key: fashion_designers_personal_id.

- **fashion_exhibitions:**
  This entity contains information about fashion exhibitions, and is also the parent entity of fashion_competitions.
  Since the hierarchy is (Partial, Exclusive) both the parent entity (i.e. fashion_exhibitions) and the child entity (i.e. fashion_competitions) are translated.
  It has the following attributes:
  - id: unique integer identifier of the fashion exhibitions.
  - duration: decimal value of the duration of the exhibition, with the format minutes.seconds.
  - location_address: string containing the address of where the exhibition is located.
  - title: string containing the title of the exhibition.
  - description: text containing a description of the exhibition.

  Primary key: id.

- **fashion_competitions:**
  This entity contains information about fashion competitions. Since it is the child of the fashion_exhibitions class, it inherits all its attributes.
  Then add the attribute:
  - year: unique integer value of the year in which the competition took place, with the format YYYY.

  Primary key: id (inherited).

- **dresses:**
  This entity represents the dresses that models can wear during the fashion exhibitions. Designers can also choose which dress to show during the fashion competitions.
  It has the following attributes:
  - id: unique integer identifier of the dress.
  - size: string containing the size of the dress.
  - colour: string containing the colour of the dress.
  - production_time: decimal value of the time required to product the dress, with the format minutes.seconds.
  - fashion_models_personal_id: reference to the personal id of the model to whom the dress is worn.

  Primary key: id.
  Foreign key: fashion_models_personal_id.
- **fabrics:**
  This entity contains information about the fabrics that have been used to make dresses or other fabrics.
  It has the following attributes:
  - id: unique integer identifier of the fabric.
  - price_per_square_meter: decimal value of the price per square meter of the fabric, with the format euro.cents and without the currency.
  - name: string containing the name of the fabric.

  The multiple attribute "property" is translated into a new table (see below).
  Private key: id.

Let's now analyse how all the relationship defined in the ER model are transformed:

- **assigned_to:**
  This relationship allows you to assign a model to a specific designer.
  Since the cardinality on the model side is (1:1) (a model is assigned to exactly one designer by assumption), this relationship is not translated and a foreign key will be added inside fashion_models, as shown before.

- **wear:**
  This relationship connects a dress to the model who wore it during fashion exhibitions and competitions.
  Since the cardinality on the dress side is (1:1) (a dress is worn by exactly one model by assumption), this relationship is not translated and a foreign key will be added inside dresses, as shown before.

- **partake_to:**
  This relationship connects the fashion models to the fashion exhibitions that they partake to.
  Since the cardinality on both side is (1:N), this relationship is translated and it contains two foreign keys.
  Primary key: the couple fashion_models_personal_id – fashion_exhibitions_id.
  Foreign keys:
  - fashion_exhibitions_id: reference to the id of the fashion exhibition.
  - fashion_models_personal_id: reference to the personal id of the model.

- **shown_in:**
  This relationship connects the dresses and the fashion competitions.
  Since the cardinality is (1:N) on the fashion competitions side and (0:1) on the dress side with low load (not all the clothes were submitted to fashion competitions, so there are a considerable number of null values), this relationship is translated and it contains two foreign keys and the attribute winner.
  Primary key: the couple dresses_id – fashion_competitions_id.
  Foreign keys:
  - dresses_id: reference to the id of the dress shown in the competition.
  - fashion_competitions_id: reference to the id of the fashion competition.

  Attribute:
  - winner: boolean value (1 or 0) representing the victory or non-victory of the dress in the fashion competition.

- **composed_of:**
  This relationship connects dresses and the respective fabrics that the dresses are made of.
  Since the cardinality is (1:N) on the dresses side and (0:N) on the fabrics, this relationship is translated and it contains two foreign keys and the attribute amount.

Primary key: the couple dresses_id – fabrics_id.
Foreign keys:

- dresses_id: reference to the id of the dress.
- fabrics_id: reference to the id of the fabric.

Attribute:

- amount: integer value of the percentage of the fabric used to realise the dress.

- **made_of:**
  This relationship connects fabrics with themselves, because a fabric can be make of another fabric.
  Since the cardinality is (0:N) on both sides, this relationship is translated and it contains two foreign keys and the attribute amount.
  Primary key: the couple fabrics_id1 – fabrics_id2.
  Foreign keys:
  - fabrics_id1: reference to fabric id made from another fabric.
  - fabrics_id2: reference to the fabric id used to make another fabric.

  Attribute:
  - amount: integer value of the percentage of the fabric used to realise the other fabric.

- **properties_of_fabrics:** This relationship connects a fabric to its properties' names.
  Primary key: the couple fabrics_id – properties_name (names of the properties).
  Foreign key:
  - fabrics_id: reference to the id of the fabric.

In conclusion, we have translated the ER model into the following tables:

- fashion_designers (personal_id, name, surname, phone_number, stage_name, career).
- fashion_models (personal_id, name, surname, phone_number, height, size, date_of_birth, fashion_designers_personal_id).
- fashion_exhibitions (id, duration, location_address, title, description).
- fashion_competitions (id, duration, location_address, title, description, year).
- dresses (id, size, colour, production_time, fashion_models_personal_id).
- fabrics (id, price_per_square_meter, name).
- partake_to (fashion_exhibitions_id, fashion_models_personal_id).
- shown_in (fashion_competitions_id, dresses_id, winner).
- composed_of (dresses_id, fabrics_id, amount).
- made_of (fabrics_id1, fabrics_id2, amount).
- properties_of_fabrics (fabrics_id, properties_name)

Legend:

- Underline attribute(s) → Primary key
- Green attribute → Foreign key

4 | Relational Model

The last step in this first part of model definition consists in the implementation of the relational model.
Taking advantage of the tables defined in the logical model, Figure 4.1 shows the final implementation of our relational model, also indicating the references of the foreign keys.

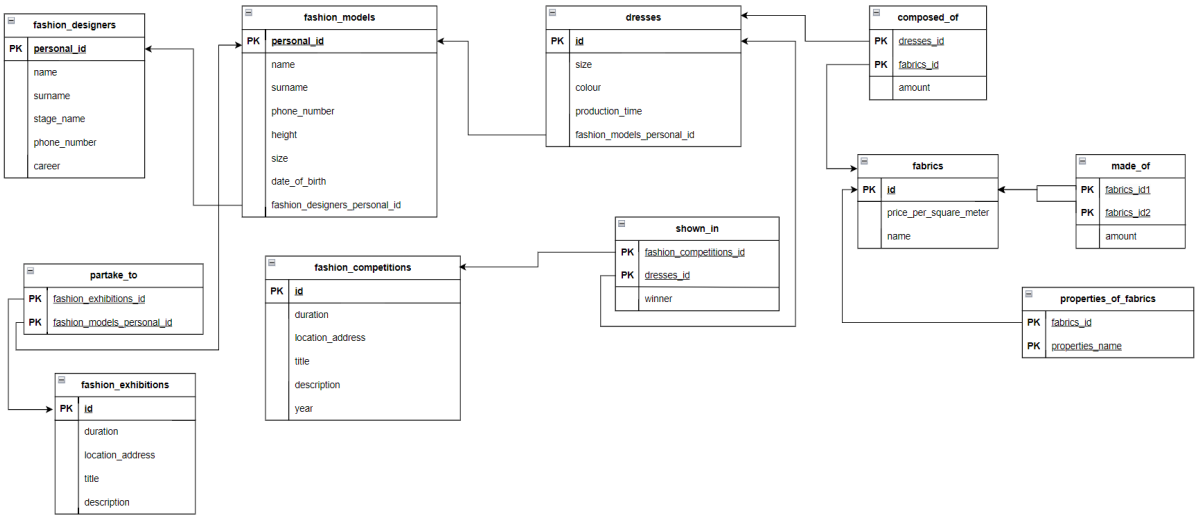

Figure 4.1: Relational model.

The tables shown follow the same indications as the tables shown in the conclusion of the logic model.
Consequently, the tables shown in Figure 4.1 have the same names, attributes, primary keys and secondary keys as the respective tables defined in the logical model.

5 | SQL requests

After having implemented the relational model of the project, it is now time to create the actual database (using the MySQL application), populate it and finally query it, analyzing the results obtained.

5.1. Tables' definition

In this chapter, we report the SQL code used to create the tables defined in the relational model.

- **Table for fashion_designers:**
  CREATE TABLE fashion_designers (

    personal_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    surname VARCHAR(255) NOT NULL,
    phone_number VARCHAR(20) NOT NULL,
    stage_name VARCHAR(255) NOT NULL,
    career TEXT NOT NULL

  );

  This SQL code creates a table called *fashion_designers*.
  The first column is *personal_id*, an integer value serving as the primary key. It is implemented with the AUTO_INCREMENT feature that automatically generates a unique ID for each record added to this table.
  The columns *name* and *surname* represent the first and last names of the designers. Both are VARCHAR(255) types.
  The *phone_number* column is a VARCHAR(20) type, storing phone numbers.
  The *stage_name* field also contains VARCHAR(255) values, which represent the "artist names" of the designers.
  The *career* column is a TEXT field, which can contain longer descriptions about

the designer's career.
All the columns have a NOT NULL constraint that will make sure every column doesn't contain NULL values.

- **Table for fashion_models:**
  CREATE TABLE fashion_models (

  personal_ID INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  surname VARCHAR(255) NOT NULL,
  phone_number VARCHAR(20) NOT NULL,
  height INT NOT NULL CHECK (height>0),
  size ENUM('XXS', 'XS', 'S', 'M', 'L', 'XL', 'XXL') NOT NULL,
  date_of_birth DATE NOT NULL,
  fashion_designers_personal_id INT NOT NULL,
  FOREIGN KEY (fashion_designers_personal_id)
  REFERENCES fashion_designers(personal_id) ON UPDATE CASCADE
  ON DELETE CASCADE

  );

  This SQL code creates a table called *fashion_models*.
  The first column is *personal_id*, an integer value serving as the primary key. It is implemented with the AUTO_INCREMENT feature that automatically generates a unique ID for each record added to this table.
  The columns *name* and *surname* represent the first and last names of the models. Both are VARCHAR(255) types.
  The *phone_number* column is a VARCHAR(20) type, storing phone numbers.
  The *height* column is of type INT and stores the height of the models. It has a CHECK (height>0) constraint, meaning only positive values are allowed.
  The *size* column uses the ENUM data type, which limits the values to a predefined set of options: 'XXS', 'XS', 'S', 'M', 'L', 'XL', and 'XXL'.
  The *date_of_birth* column has DATE type and stores the model's birth date.
  The *fashion_designers_personal_id* column is an INT columns that serves as a foreign key, linking each model to the assigned fashion designer from the *fashion_designers* table, by referencing to the *personal_id* column.
  The ON UPDATE CASCADE and ON DELETE CASCADE clause ensure that

changes to the designer's *personal_ id* will automatically update the corresponding records in the *fashion_ models* table.
All the columns have a NOT NULL constraint that will make sure every column doesn't contain NULL values.

- **Table for fashion_exhibitions:**
CREATE TABLE fashion_exhibitions (

    id INT PRIMARY KEY AUTO_INCREMENT,
    duration DECIMAL(5,2) NOT NULL CHECK (duration>0),
    location_address VARCHAR(255) NOT NULL,
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL

);

This SQL code creates a table named *fashion_ exhibitions*.
The first column is *id*, an integer value serving as the primary key. It is implemented with the AUTO_INCREMENT feature that automatically generates a unique ID for each record added to this table.
The *duration* column is of type DECIMAL(5,2), which allows the insertion of numerical values with up to 5 digits in total, and 2 of those can be after the decimal point. The CHECK (duration>0) constraint ensures that the value entered is greater than 0.
The *location_ address* column is a VARCHAR(255) field, allowing to save the address of the exhibition's location.
The *title* column is also a VARCHAR(255) field that saves the title of the exhibition.
The *description* column is of type TEXT, which is used for longer description of the exhibition.
All the columns have a NOT NULL constraint that will make sure every column doesn't contain NULL values.

- **Table for fashion_competitions:**
CREATE TABLE fashion_competitions (

    id INT PRIMARY KEY AUTO_INCREMENT,
    duration DECIMAL(5,2) NOT NULL CHECK (duration>0),

```
    location_address VARCHAR(255) NOT NULL,
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    year YEAR UNIQUE NOT NULL
);
```

This SQL code creates a table named *fashion_competitions*.
The first column is *id*, an integer value serving as the primary key. It is implemented with the AUTO_INCREMENT feature that automatically generates a unique ID for each record added to this table.
The *duration* column is of type DECIMAL(5,2), which allows the insertion of numerical values with up to 5 digits in total, and 2 of those can be after the decimal point. The CHECK (duration>0) constraint ensures that the value entered is greater than 0.
The *location_address* column is a VARCHAR(255) field, allowing to save the address of the competition's location.
The *title* column is also a VARCHAR(255) field that saves the title of the competition.
The *description* column is of type TEXT, which is used for longer description of the competition.
The *year* column has type YEAR and is used for saving the year of the competitions.
All the columns have a NOT NULL constraint that will make sure every column doesn't contain NULL values.

- **Table for dresses:**
  CREATE TABLE dresses (

```
    id INT PRIMARY KEY AUTO_INCREMENT,
    size ENUM('UNI', 'XXS', 'XS', 'S', 'M', 'L', 'XL', 'XXL') NOT NULL,
    colour VARCHAR(50) NOT NULL,
    production_time DECIMAL(5,2) NOT NULL CHECK (production_time>0),
    fashion_models_personal_id INT NOT NULL,
    FOREIGN KEY (fashion_models_personal_id)
    REFERENCES fashion_models(personal_id) ON UPDATE CASCADE
    ON DELETE CASCADE
```

);

This SQL code creates a table named *dresses*.
The first column is *id*, an integer value serving as the primary key. It is implemented with the AUTO_INCREMENT feature that automatically generates a unique ID for each record added to this table.
The *size* column uses the ENUM data type, which restricts the values to a specific set of predefined options: 'UNI', 'XXS', 'XS', 'S', 'M', 'L', 'XL', and 'XXL'.
The *colour* column is a VARCHAR(50) field that allows the insertion of the colours of the dresses.
The *production_time* column is of type DECIMAL(5,2), which allows the insertion of numerical values with up to 5 digits in total, and 2 of those can be after the decimal point. The CHECK (production_time>0) constraint ensures that the value entered is greater than 0.
The *fashion_models_personal_id* column is an INT columns that serves as a foreign key, linking each dress to the models who wore it by referencing to the *personal_id* column of the table *fashion_models*.
The ON UPDATE CASCADE and ON DELETE CASCADE clause ensure that changes to the model's *personal_id* will automatically update the corresponding records in the *dresses* table.
All the columns have a NOT NULL constraint that will make sure every column doesn't contain NULL values.

- **Table for fabrics:**
  CREATE TABLE fabrics (

    id INT PRIMARY KEY AUTO_INCREMENT,
    price_per_square_meter DECIMAL(10,2) NOT NULL CHECK
    (price_per_square_meter>0),
    name VARCHAR(255) NOT NULL

  );

  This SQL code creates a table named *fabrics*.
  The first column is *id*, an integer value serving as the primary key. It is implemented with the AUTO_INCREMENT feature that automatically generates a unique ID for each record added to this table.

The *price_per_square_meter* column is of type DECIMAL(10,2), which allows the insertion of numerical values with up to 10 digits in total, and 2 of those can be after the decimal point. The CHECK (price_per_square_meter>0) constraint ensures that the value entered is greater than 0.
The *name* column is a VARCHAR(255) field, which allows to save the name of the fabrics.
All the columns have a NOT NULL constraint that will make sure every column doesn't contain NULL values.

- **Table for partake_to (Relationship between Fashion Exhibitions and Fashion Models):**
  CREATE TABLE partake_to (

    fashion_exhibitions_id INT,
    fashion_models_personal_id INT,
    PRIMARY KEY (fashion_exhibitions_id,
    fashion_models_personal_id),
    FOREIGN KEY (fashion_exhibitions_id)
    REFERENCES fashion_exhibitions(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (fashion_models_personal_id)
    REFERENCES fashion_models(personal_id) ON UPDATE CASCADE ON DELETE CASCADE

  );

  This SQL code creates a table named *partake_to*.
  fashion_exhibitions_id and *fashion_models_personal_id* are two columns of type INT that act as the primary key of this table.
  The two columns are also foreign keys for the *fashion_exhibitions* and *fashion_models* tables respectively, referring to the exhibition's *id* and the model's *personal_id* attributes.
  The ON UPDATE CASCADE and ON DELETE CASCADE clauses ensure that changes to the referred tables will automatically update the corresponding records in the *partake_to* table.

- **Table for shown_in (Relationship between Fashion Competitions and Dresses):**
  CREATE TABLE shown_in (

    fashion_competitions_id INT,
    dresses_id INT,
    winner BOOLEAN NOT NULL,
    PRIMARY KEY (fashion_competitions_id, dresses_id),
    FOREIGN KEY (fashion_competitions_id)
    REFERENCES fashion_competitions(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (dresses_id) REFERENCES dresses(id) ON UPDATE CASCADE ON DELETE CASCADE

  );

  This SQL code creates a table named *shown_in*.
  fashion_competitions_id and *dresses_id* are two columns of type INT that act as the primary key of this table.
  The two columns are also foreign keys for the *fashion_competitions* and *dresses* tables respectively, referring to the competition's *id* and the dress' *id* attributes.
  The ON UPDATE CASCADE and ON DELETE CASCADE clauses ensure that changes to the referred tables will automatically update the corresponding records in the *shown_in* table.
  The *winner* column has type BOOLEAN, which allows the storage of only 1 or 0 values. It is also marked as NOT NULL, ensuring the presence of all values in this column.

- **Table for composed_of (Relationship between Dresses and Fabrics):**
  CREATE TABLE composed_of (

    dresses_id INT,
    fabrics_id INT,
    amount INT NOT NULL CHECK (amount>0 AND amount<=100),
    PRIMARY KEY (dresses_id, fabrics_id),
    FOREIGN KEY (dresses_id) REFERENCES dresses(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (fabrics_id) REFERENCES fabrics(id) ON UPDATE

CASCADE ON DELETE CASCADE

);

This SQL code creates a table named *composed_ of*.
dresses_ id and *fabrics_ id* are two columns of type INT that act as the primary key of this table.
The two columns are also foreign keys for the *dresses* and *fabrics* tables respectively, referring to the dress' *id* and the fabric's *id* attributes.
The ON UPDATE CASCADE and ON DELETE CASCADE clauses ensure that changes to the referred tables will automatically update the corresponding records in the *composed_ of* table.
The *amount* column has INT type and allows to specifies the quantity of the fabric used for each dress. The CHECK (amount>0 AND amount<=100) constraint ensures that the amount entered is between 0 (not included) and 100 (included). It is also marked as NOT NULL, ensuring the presence of all values in this column.

- **Table for made_of (Self-Referencing Relationship between Fabrics):**
  CREATE TABLE made_of (

  fabrics_id1 INT,
  fabrics_id2 INT,
  amount INT NOT NULL CHECK (amount>0 AND amount<=100),
  PRIMARY KEY (fabrics_id1, fabrics_id2),
  FOREIGN KEY (fabrics_id1) REFERENCES fabrics(id) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (fabrics_id2) REFERENCES fabrics(id) ON UPDATE CASCADE ON DELETE CASCADE

  );

  This SQL code creates a table named *made_ of*.
  fabrics_ id1 and *fabrics_ id2* are two columns of type INT that act as the primary key of this table.
  The two columns are also foreign keys for the same *fabrics* table, referring to the two fabrics' *id* attributes.
  The ON UPDATE CASCADE and ON DELETE CASCADE clauses ensure that

changes to the referred table will automatically update the corresponding records in the *made_of* table.
The *amount* column has INT type and allows to specifies the quantity of the fabric used to make another fabric. The CHECK (amount>0 AND amount<=100) constraint ensures that the amount entered is between 0 (not included) and 100 (included). It is also marked as NOT NULL, ensuring the presence of all values in this column.

- **Table for properties_of_fabrics (Relationship between Fabrics and Properties):**
  CREATE TABLE properties_of_fabrics (

    fabrics_id INT,
    properties_name VARCHAR(255),
    PRIMARY KEY (fabrics_id, properties_name),
    FOREIGN KEY (fabrics_id)
    REFERENCES fabrics(id) ON UPDATE CASCADE ON
    DELETE CASCADE

  );

  This SQL code creates a table named *properties_of_fabrics*.
  fabrics_id and *properties_name* are the two columns used as primary key, which uniquely identify the table.
  fabrics_id has an INT type and it is a foreign key for the *fabrics* table, referring to the fabric's *id* attribute.
  The ON UPDATE CASCADE and ON DELETE CASCADE clause ensure that changes to the referred table will automatically update the corresponding records in the *properties_of_fabrics* table.
  The *properties_name* column has type VARCHAR(255) and it allows to store the names of the properties of the fabrics.

5.2. Database population

In this chapter, we will see how the values used to populate the tables were generated.

5.2.1. ChatGPT and Python

For data generation, we decided to use ChatGPT instead of "mockaroo". The use of generative AI allows for better content creation, in which the chatbot excels. Furthermore, it is able to create a brief description of the designer's career and the exhibitions.

We then used Python scripts to guarantee that the relationships were correct; for example, for the relation "partake_to" that connects models to exhibitions, we first assigned one model to every exhibition, and to every model, one exhibition, and then assigned more random models to exhibitions. This methodology ensures the relationship is 1:N for exhibitions to models and 1:N for models to exhibitions.
The same procedure was used to generate the data of the other tables representing relationships, so as to guarantee the correctness of the references of the foreign keys.

Lastly, we used Python also to check that the measure of the model corresponds to the actual measure of the assigned dress.

5.2.2. Data Screenshots

We now report some short screenshots of the data that populate the tables (not all the data in the tables is shown, otherwise the photos would be excessively large).

personal_id	name	surname	phone_number	stage_name	career
1	Sophia	Mason	+44 793 2341234	Avant Couture	Sophia has made a name for herself in London with her bold, futuristic design...
2	James	Harrison	+1 202 5559876	Timeless Chic	James has been a staple in the fashion industry for over two decades. His de...
3	Isabella	Smith	+1 213 5551234	Urban Visionary	Isabella is known for blending high fashion with urban culture. Her work has ...
4	Olivia	Clark	+61 400 123456	Boho Glam	Olivia's designs bring together bohemian style and glamour in a unique way. ...
5	Benjamin	Johnson	+1 415 5556543	Minimal Edge	Benjamin is a minimalist designer with a focus on sharp, clean lines. His desig...
6	Amelia	Davies	+44 770 9876543	Eco Haute	Amelia is at the forefront of sustainable fashion, using only recycled material...
7	William	Jones	+61 420 5558765	Modern Maverick	William is a rebellious designer known for his daring menswear collections. He ...
8	Charlotte	Miller	+1 305 5553210	Ethereal Dream	Charlotte's collections are known for their dreamy, ethereal quality. She uses...
9	Ethan	Taylor	+1 718 5553321	Street Pulse	Ethan draws inspiration from street culture and music. His collections often fe...
10	Mia	Thomas	+44 790 5559901	Classic Revival	Mia brings a fresh take on classic fashion, reviving vintage styles. Her design...
11	Henry	Wright	+1 202 5558745	Dapper Tailor	Henry is a master of tailoring, creating elegant and sophisticated suits. He bl...
12	Ava	Martinez	+34 611 5553422	Floral Essence	Ava's designs are heavily influenced by nature, particularly flowers. Her colle...
13	Noah	Evans	+1 312 5557612	Bold Form	Noah is celebrated for his bold, architectural designs. He often experiments ...
14	Emma	Walker	+44 775 5550987	Vintage Twist	Emma combines vintage fashion with modern trends in an innovative way. He...
15	Lucas	White	+1 310 5557841	Dark Allure	Lucas specializes in dark, mysterious aesthetics. His collections often feature ...
16	Emily	Hill	+44 789 5556721	Silk Whispers	Emily's designs are known for their softness and flowing silhouettes. She ofte...
17	Alexander	Green	+1 646 5559210	Urban Edge	Alexander has made a name for himself by blending streetwear with high fas...

Figure 5.1: fashion_designers

personal_ID	name	surname	phone_number	height	size	date_of_birth	fashion_designers_personal_id
1	Joel	Patel	+44 458 8353811	190	XXL	1997-09-08	1
2	Katie	Underwood	+44 505 8381583	167	S	2006-05-26	2
3	Jeffery	Mcgrath	+44 342 4269810	181	L	1991-06-13	3
4	Diane	Lopez	+44 392 6707787	189	L	1991-10-05	4
5	Roy	Conner	+44 745 5881484	190	XXL	2006-05-28	5
6	Stephanie	Dodson	+44 738 2299572	173	XXL	1993-07-22	6
7	Robert	Washington	+44 674 2577091	189	XXS	2002-06-09	7
8	Frederick	Wright	+44 208 1824821	172	L	1993-07-04	8
9	Marcus	Morgan	+44 333 5545952	193	XL	1999-09-11	9
10	Zachary	Rose	+44 689 5448255	168	XXL	2001-11-20	10
11	Brittany	Reed	+44 779 5187091	195	M	1994-04-14	11
12	Timothy	Brown	+44 968 9519077	166	XXL	1992-03-08	12
13	Anthony	Gray	+44 441 2536202	184	XS	2005-01-04	13
14	Amy	Mack	+44 871 8530664	184	XXL	1995-01-12	14
15	Eric	Carpenter	+44 966 9952257	173	L	1999-01-09	15
16	Paul	Taylor	+44 625 8602765	171	XS	1993-09-06	16

Figure 5.2: fashion_models

id	duration	location_address	title	description
1	60.00	101 Style St, Milan, Italy	Milan Fashion Extravaganza	Join us for an unforgettable evening of style an...
2	75.50	102 Fashion Ave, New York, NY 10001	New York Spring Showcase	Celebrate the arrival of spring with this vibrant ...
3	90.25	103 Chic Blvd, Paris, France	Paris Fashion Gala	Experience the glamour of Parisian fashion at th...
4	80.00	104 Trendy Ln, London, UK	London Style Festival	Join us for a celebration of British fashion and s...
5	95.00	105 Elegant Rd, Tokyo, Japan	Tokyo Fashion Odyssey	Explore the diverse world of fashion at this exci...
6	70.75	106 Glamour St, Barcelona, Spain	Barcelona Fashion Week	Immerse yourself in the vibrant atmosphere of ...
7	100.50	107 Artistic Blvd, Berlin, Germany	Berlin Fashion Revolution	Witness a revolution in fashion with groundbrea...
8	85.00	108 Stylish Rd, Sydney, Australia	Sydney Fashion Voyage	Join us for a journey through the latest trends i...
9	60.25	109 Unique St, Toronto, Canada	Toronto Fashion Showcase	Discover the best of Canadian fashion in this ex...
10	75.00	110 Trendy Ave, Milan, Italy	Milan Trend Setters	Explore the latest trends and styles in the heart...
11	90.00	111 Couture Rd, New York, NY 10001	New York Fashion Discovery	Experience the excitement of discovering new f...
12	80.50	112 Fashion St, Paris, France	Paris Fashion Impressions	Celebrate the beauty of fashion in Paris with thi...
13	95.25	113 Stylish Ave, London, UK	London Fashion Showcase	Witness the latest trends in London fashion. Thi...
14	70.50	114 Trendy Rd, Tokyo, Japan	Tokyo Style Exhibition	Experience the vibrant and dynamic styles of T...
15	85.00	115 Glamour Ln, Barcelona, Spain	Barcelona Chic Show	Join us for a celebration of chic fashion in Barcel...
16	100.25	116 Elegant Blvd, Berlin, Germany	Berlin Fashion Perspectives	Explore diverse perspectives in fashion at this c...

Figure 5.3: fashion_exhibitions

id	duration	location_address	title	description	year
1	120.00	1 Fashion St, New York, NY 10001	The Elite Runway	Witness the most exclusive fashion competition ...	2024
2	90.50	2 Trendy Rd, Los Angeles, CA 90001	Summer Style Showdown	Join us for an exciting competition where summ...	2023
3	75.25	3 Chic Ave, Paris, France	Paris Couture Competition	This prestigious event highlights the best in hau...	2022
4	110.75	4 Vogue St, Milan, Italy	Milan Fashion Championship	Witness emerging talents as they present their ...	2021
5	85.00	5 Stylish Blvd, London, UK	London Fashion Face-Off	The best of British fashion is here! Watch desig...	2020
6	95.60	6 Glamour St, Tokyo, Japan	Tokyo Trendsetters	Experience cutting-edge fashion from the most i...	2019
7	100.50	7 Elegant Way, Barcelona, Spain	Barcelona Fashion Fiesta	Celebrate vibrant designs and cultural expressi...	2018
8	65.00	8 Modern Ln, Sydney, Australia	Sydney Fashion Showcase	Catch the latest trends from Australian designe...	2017
9	80.25	9 Unique Rd, Toronto, Canada	Toronto Fashion Challenge	Discover emerging talents in the Canadian fashi...	2016
10	70.10	10 Artistic Blvd, Berlin, Germany	Berlin Fashion Innovation	Witness the future of fashion as designers pres...	2015
11	130.00	11 Royal St, Paris, France	Paris Fashion Festival	Join us for a spectacular display of talent and cr...	2014
12	95.50	12 Heritage Ln, Milan, Italy	Milan Vintage Showdown	Explore the best of vintage fashion in this thrilli...	2013
13	85.75	13 Couture Blvd, Tokyo, Japan	Tokyo Street Style Contest	Celebrate the vibrant street fashion of Tokyo. ...	2012
14	100.00	14 Fashion Ave, London, UK	British Fashion Awards	Join us for a glamorous evening celebrating the ...	2011
15	90.00	15 Chic St, Sydney, Australia	Sydney Eco Fashion Show	Experience the fusion of fashion and sustainabil...	2010
16	105.50	16 Trendy Rd, New York, NY 10001	New York Fashion Week C...	This iconic event brings together top designers ...	2009
17	80.00	17 Stylish Ave, Barcelona, Spain	Barcelona Fashion Evolution	Explore the evolution of fashion through this ex...	2008
18	95.75	18 Trendy St, Berlin, Germany	Berlin Fashion Night	Join us for a night celebrating the vibrant fashio...	2007
19	110.50	19 Elegant Rd, Toronto, Canada	Toronto Fashion Spectrum	Witness a colorful array of designs from Canadi...	2006
20	70.00	20 Fashion Way, Milan, Italy	Milan Style Showcase	Experience the elegance of Milanese fashion. T...	2005

Figure 5.4: fashion_competitions

id	size	colour	production_time	fashion_models_personal_id
1	XXL	MediumVioletRed	166.64	1
2	S	Purple	239.03	2
3	L	RosyBrown	173.59	3
4	L	PeachPuff	146.76	4
5	XXL	BlanchedAlmond	195.63	5
6	XXL	PeachPuff	102.75	6
7	XXS	MediumPurple	124.66	7
8	UNI	Lavender	67.40	8
9	XL	CadetBlue	248.39	9
10	XXL	Tan	223.89	10
11	M	Sienna	293.10	11
12	XXL	Pink	68.47	12
13	XS	DarkOrange	65.94	13
14	XXL	LimeGreen	279.43	14
15	UNI	OrangeRed	146.49	15
16	XS	MistyRose	256.97	16
17	S	FloralWhite	137.47	17
18	S	SlateBlue	283.80	18
19	UNI	Green	226.88	19

Figure 5.5: dresses

id	price_per_square_meter	name
1	15.99	Cotton Poplin
2	25.50	Silk Satin
3	8.75	Polyester Blend
4	12.30	Linen
5	22.99	Wool Tweed
6	30.00	Bamboo Fabric
7	10.20	Denim
8	18.80	Viscose
9	14.50	Rayon Challis
10	35.00	Cashmere
11	20.75	Taffeta
12	5.99	Jersey Knit
13	9.50	Canvas
14	17.85	Gabardine
15	40.00	Silk Organza
16	11.15	Cotton Twill
17	7.25	Microfiber
18	13.45	Crêpe
19	33.30	Brocade

Figure 5.6: fabrics

fashion_exhibitions_id	fashion_models_personal_id
11	1
31	1
33	1
30	2
40	2
22	3
47	3
49	3
2	4
10	4
33	4
43	4
46	4
9	5
21	5
9	6

Figure 5.7: partake_to

fashion_competitions_id	dresses_id	winner
1	1	1
1	2	0
1	3	0
1	4	0
1	5	0
1	6	0
1	7	0
2	8	1
2	9	0
2	10	0
2	11	0
2	12	0
2	13	0
3	14	1
3	15	0
3	16	0
3	17	0

Figure 5.8: shown_in

	dresses_id	fabrics_id	amount
▶	1	11	3
	1	17	97
	2	29	22
	2	39	78
	3	44	100
	4	22	54
	4	30	46
	5	9	37
	5	10	36
	5	49	27
	6	9	20
	6	25	79
	6	34	1
	7	20	82
	7	41	18
	8	1	83
	8	11	17
	9	7	60
	9	13	16
	9	31	24

Figure 5.9: composed_of

fabrics_id1	fabrics_id2	amount
1	2	60
1	3	40
4	5	75
4	6	25
7	9	30
7	10	70
8	10	50
8	12	30
8	13	20
11	12	40
11	13	30
11	16	30
14	23	20
14	24	80
15	22	40
15	24	60

Figure 5.10: made_of

fabrics_id	properties_name
1	Breathable
1	Lightweight
1	Soft
2	Luxurious
2	Shiny
2	Smooth
3	Durable
3	Wrinkle-Resistant
4	Absorbent
4	Cool to the Touch
4	Warm
5	Warm
5	Water-Resistant
6	Lightweight
6	Stretchable
7	Heavyweight
7	Reinforced

Figure 5.11: properties_of_fabrics

5.3. Queries

In this chapter, we will propose a series of queries with different clauses, with the final aim of querying the database and the different tables, then analyzing the result obtained.

(For queries that result in a table with many rows, only some of those rows are reported.)

QUERY 1: WHERE
Request:

Find the title and id of fashion exhibitions that last more than 90 minutes.

Resolution:

```
SELECT title, id
FROM fashion_exhibitions
WHERE duration > 90
```

Description:

This query retrieves the titles and IDs of fashion exhibitions from the *fashion_exhibitions* table where the duration of each exhibition is greater than 90 minutes.

Output:

title	id
Paris Fashion Gala	3
Tokyo Fashion Odyssey	5
Berlin Fashion Revolution	7
London Fashion Showcase	13
Berlin Fashion Perspectives	16
Toronto Fashion Experience	21
London Style Revolution	22
Tokyo Fashion Fusion	23
Paris Fashion Gala	28
London Fashion Inspirations	30
Milan Fashion Voyage	36
Toronto Fashion Parade	37
Milan Fashion Showcase	43
Tokyo Fashion Festival	44
Paris Fashion Showcase	49
Milan Fashion Odyssey	50

QUERY 2: WHERE, LIMIT, LIKE
Request:

Find the information of the five tallest models wearing a size containing 'S'.

Resolution:

```
SELECT *
FROM fashion_models
WHERE size LIKE "%S"
ORDER BY height DESC
LIMIT 5;
```

Description:

This query retrieves the information of the top five models from *fashion_ models* table, but keep only the models whose size contain an S at the end.
It uses limit and order by to obtain only the 5 tallest models.

Output:

personal_ID	name	surname	phone_number	height	size	date_of_birth	fashion_designers_personal_id
59	Christian	Alexander	+44 233 1456105	195	S	1992-09-30	31
125	Ruben	Johnson	+44 767 5105416	195	XS	2005-10-17	43
134	Jason	Mcgee	+44 746 7489001	193	XS	1992-12-17	9
25	Brandon	Martinez	+44 325 3464970	193	XXS	1995-04-21	25
142	Hannah	Rivera	+44 555 4361651	193	XS	1997-01-12	14
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

QUERY 3: WHERE, IN, Nested Query
Request:

Find name, surname and id of models assigned to designers with stage name: 'Avant Couture', 'Boho Glam', 'Bohemian Luxe'.

Resolution:

```
SELECT m.name AS model_name, m.surname AS model_surname, m.personal_id
FROM fashion_models as m
WHERE m.fashion_designers_personal_id IN (
    SELECT d.personal_id
    FROM fashion_designers AS d
    WHERE d.stage_name IN ('Avant Couture', 'Boho Glam','Bohemian Luxe')
);
```

Description:

This query retrieves the names, surnames, and personal IDs of fashion models from the *fashion_ models* table. It includes only those models that are assigned with fashion designers whose stage names are 'Avant Couture', 'Boho Glam', or 'Bohemian Luxe'. This is done by using a subquery, which is used to retrieve from the *fashion_ designers* table the personal IDs of the specified fashion designers.

Output:

model_name	model_surname	personal_id
Joel	Patel	1
Jose	Jones	66
Austin	Pearson	82
Kyle	Davidson	107
Diane	Lopez	4
Eric	Nicholson	60
Ryan	Cooper	64
Michael	Collins	128
Jacob	Mdaughlin	48
David	Kelly	86

QUERY 4: GROUP BY, 1 JOIN, AS
Request:

Show the id, title and number of dresses shown for each fashion competitions.

Resolution:

```
SELECT sin.fashion_competitions_id, comp.title, count(*) AS number_dresses_shown
FROM shown_in AS sin JOIN fashion_competitions AS comp
     ON sin.fashion_competitions_id = comp.id
GROUP BY sin.fashion_competitions_id, comp.title
```

Description:

The query retrieves the IDs, titles and the count of dresses displayed for each fashion competition by joining the *fashion_ competitions* and *shown_ in* tables based on the competition IDs. It groups the results by the competition's ID and title, and uses the count(*) function to calculate the total number of dresses shown in each competition.

Output:

fashion_competitions_id	title	number_dresses_shown
1	The Elite Runway	7
2	Summer Style Showdown	6
3	Paris Couture Competition	7
4	Milan Fashion Championship	7
5	London Fashion Face-Off	7
6	Tokyo Trendsetters	6
7	Barcelona Fashion Fiesta	7
8	Sydney Fashion Showcase	6
9	Toronto Fashion Challenge	6
10	Berlin Fashion Innovation	6
11	Paris Fashion Festival	6
12	Milan Vintage Showdown	5
13	Tokyo Street Style Contest	6
14	British Fashion Awards	5
15	Sydney Eco Fashion Show	5
16	New York Fashion Week C...	5
17	Barcelona Fashion Evolution	5

QUERY 5: WHERE, GROUP BY
Request:

Find the number of occurrences of each dresses sizes, but only for dresses with a production time greater than 60 minutes.

Resolution:

```
SELECT size, count(*) as occurrence
FROM dresses
WHERE production_time > 60
GROUP BY size
```

Description:

This query returns as output the number of occurrences for each dress size, but only for dresses with a production time over 60 minutes. It selects the size and counts the number of occurrences for each one, filtering the results based on if the production time is greater than 60 minutes.

Output:

size	occurrence
XXL	58
S	57
L	47
XXS	48
UNI	109
XL	35
M	53
XS	43

QUERY 6: GROUP BY, HAVING, AS
Request:

Find fashion models who have participated in five or more fashion exhibitions.
Shows the fashion models id and number of participations (as number_participations).

Resolution:

```
SELECT  fashion_models_personal_id, COUNT(*) as number_participations
FROM partake_to
GROUP BY fashion_models_personal_id
HAVING COUNT(*) >= 5
```

Description:

This query return as output the list of fashion models who have participated in five or more fashion exhibitions. It selects the personal ID of each model and counts their total number of participations, filtering the results to include only those with five or more participations. It is shown the model's personal ID along with the corresponding number of participations in fashion exhibitions.

Output:

fashion_models_personal_id	number_participations
4	5
8	5
9	5
10	5
23	5
25	6
28	5
30	6
36	5
39	5
41	5
42	5
44	5
46	5
65	5
67	5
74	5
94	8
100	5
114	5
121	6
123	5
125	6
136	6
146	7

QUERY 7: WHERE, GROUP BY, HAVING, AS
Request:

Find the average height of the models born from year 2000, group by the year of birth and having the height average of the group bigger than 180cm. Show also the number of models in each group.

Resolution:

```
SELECT YEAR(fashion_models.date_of_birth) AS birth_year,
       AVG(fashion_models.height) AS avg_height,
       COUNT(*) AS total_models
FROM fashion_models
WHERE YEAR(fashion_models.date_of_birth) >= 2000
GROUP BY YEAR(fashion_models.date_of_birth)
HAVING AVG(fashion_models.height) > 180;
```

Description:

This query returns the average height of fashion models born from the year 2000 onward, grouping the results by their birth year. It includes only the groups where the average height exceeds 180 cm. In addition, it shows the total number of models in each group and provide the birth year, the average height and the model count for the eligible groups.

Output:

	birth_year	avg_height	total_models
▶	2006	181.0000	3
	2005	180.7273	11
	2004	186.6667	3

QUERY 8: WHERE, Nested Query, GROUP BY
Request:

Find the locations where fashion exhibitions have been held and their respective number, but only those locations that have not hosted fashion competitions.

Resolution:

```
SELECT location_address, count(id) as exhibitions
FROM fashion_exhibitions
WHERE location_address NOT IN(
    SELECT DISTINCT location_address
    FROM fashion_competitions
    )
GROUP BY location_address
```

Description:

This query retrieves the addresses of locations where fashion exhibitions have been held, along with the number of exhibitions at each location. It excludes any locations that have hosted fashion competitions by using a subquery that selects distinct addresses from the *fashion_competitions* table. The output shows only the locations where no fashion competitions were held, and the corresponding count of exhibitions at each location.

Output:

location_address	exhibitions
101 Style St, Milan, Italy	1
102 Fashion Ave, New York, NY 10001	1
103 Chic Blvd, Paris, France	1
104 Trendy Ln, London, UK	1
105 Elegant Rd, Tokyo, Japan	1
106 Glamour St, Barcelona, Spain	1
107 Artistic Blvd, Berlin, Germany	1
108 Stylish Rd, Sydney, Australia	1
109 Unique St, Toronto, Canada	1
110 Trendy Ave, Milan, Italy	1
111 Couture Rd, New York, NY 10001	1
112 Fashion St, Paris, France	1
113 Stylish Ave, London, UK	1
114 Trendy Rd, Tokyo, Japan	1
115 Glamour Ln, Barcelona, Spain	1

QUERY 9: WHERE, GROUP BY, HAVING, 1 JOIN

Request:

> Find the fabrics' names, fabrics' prices per square meter and the total counter of dresses for each fabrics. Select only the dresses whose amount of fabric used is more than 75% and the price per square meter of the fabric is above 25.

Resolution:

```
SELECT fabrics.name AS fabric_name, fabrics.price_per_square_meter,
       COUNT(composed_of.dresses_id) AS total_dresses
FROM composed_of JOIN fabrics ON composed_of.fabrics_id = fabrics.id
WHERE amount > 75 and price_per_square_meter > 25
GROUP BY fabrics.name, fabrics.price_per_square_meter
HAVING count(composed_of.dresses_id) >= 5;
```

Description:

> This query retrieves the fabric names, their price per square meter, and the total number of dresses that use each fabric. It filters the results to include only fabrics where more than 75% are used in the dresses, and the price per square meter of the fabric is greater than 25. In addition,it ensures that only fabrics used in at least five different dresses are displayed.

Output:

fabric_name	price_per_square_meter	total_dresses
Modal	29.99	6
Silk Habotai	30.20	5
Silk Satin	25.50	5
Hemp Fabric	27.50	8
Cashmere	35.00	5
Satin Stretch	34.90	9
Bamboo Fabric	30.00	7
Satin Backed Crepe	28.50	6
Brocade	33.30	6
Lace	36.40	5

QUERY 10: WHERE, GROUP BY, HAVING, 2 JOINs

Request:

> Find the ID, name and surname of the fashion designers who have designed at least 3 dresses with size “UNI”.

Resolution:

```
SELECT fd.personal_id, fd.name, fd.surname, COUNT(*) as number_UNI_dresses
FROM (fashion_designers as fd JOIN fashion_models as fm ON fd.personal_id = fm.fashion_designers_personal_id)
     JOIN dresses as d ON fm.personal_id = d.fashion_models_personal_id
WHERE d.size = "UNI"
GROUP BY fd.personal_id, fd.name, fd.surname
HAVING COUNT(*) >= 3
```

Description:

> This query shows the personal ID, name, and surname of fashion designers who have designed at least three dresses with size "UNI". It joins the *fashion_designers* table with the *dresses* table via the *fashion_models* table, and filters the results to include only those designers who have designed three or more dresses of size "UNI".

Output:

personal_id	name	surname	number_UNI_dresses
8	Charlotte	Miller	6
15	Lucas	White	7
19	Daniel	Adams	3
29	Dylan	Campbell	4
33	Jacob	Stewart	7
34	Layla	Bell	3
45	Jackson	Cook	7
30	Avery	Young	8
3	Isabella	Smith	3
4	Olivia	Clark	4
46	Avery	Foster	4
7	William	Jones	3
32	Chloe	Parker	10
16	Emily	Hill	3
27	Logan	Ramirez	4
43	Caleb	Ward	6
14	Emma	Walker	3
31	Jack	Turner	3
5	Benjamin	Johnson	4

6 | Conclusion

The first Enterprise ICT Architecture project consisted of defining an ER, logical and relational model starting from a text to be analysed.
We then defined the respective tables on MySQL, generated and populated values to test the correctness of 10 queries that we formulated, finally verifying the correctness of the results.
The project was very stimulating and we learned several fundamental concepts for defining and creating relationship databases.