|  |
| --- |
|  |
| Course Work  EHU, SQL and Data Processing |

**Created by:  
Kanstatnsin Loseu**

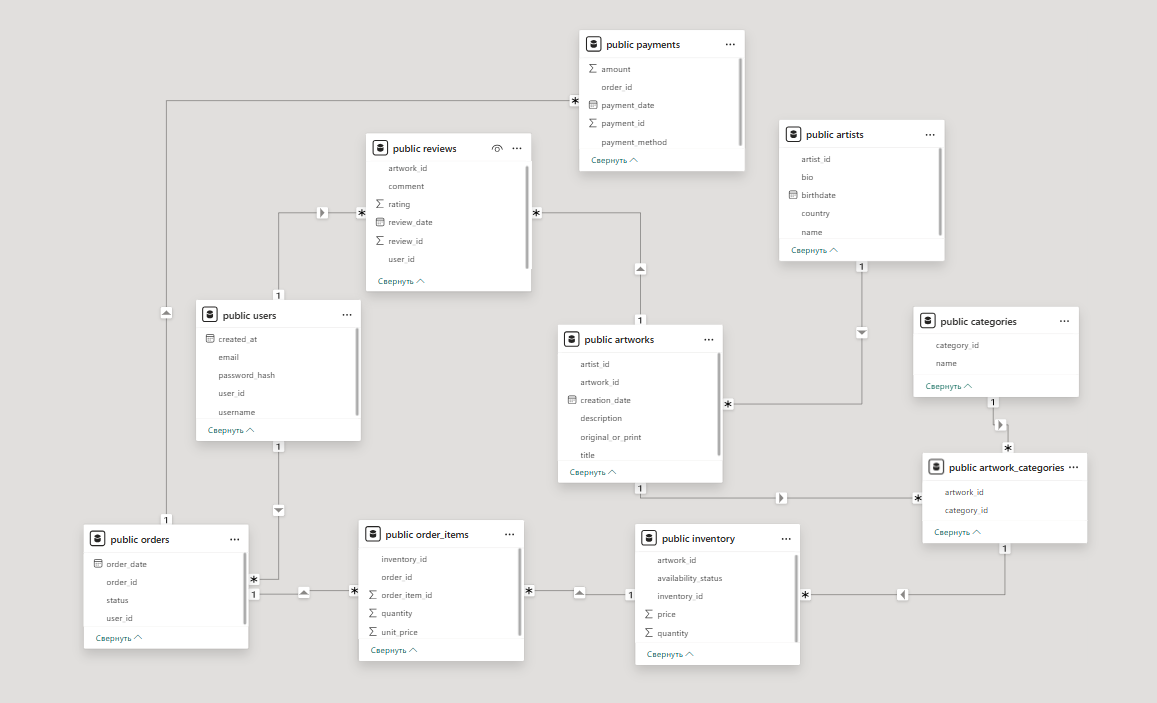
**Topic:**

**"Art prints or original artworks."**

### Part 1: Design and develop all needed DB objects to support functionality of your Application. OLTP Schema:

* Contains 10 normalized tables supporting detailed transactional operations.
* Key entities include:
* users, artists, artworks, orders, order\_items, payments, reviews,
* and metadata tables like categories, artwork\_categories, and Primary keys uniquely identify records;
* foreign keys ensure referential integrity.
* For example: Artworks are linked to artists via artist\_id, orders are linked to users, reviews reference both artworks and users
* Relationships are primarily one-to-many (e.g., one user → many orders), with one many-to-many link (artworks<->categories).

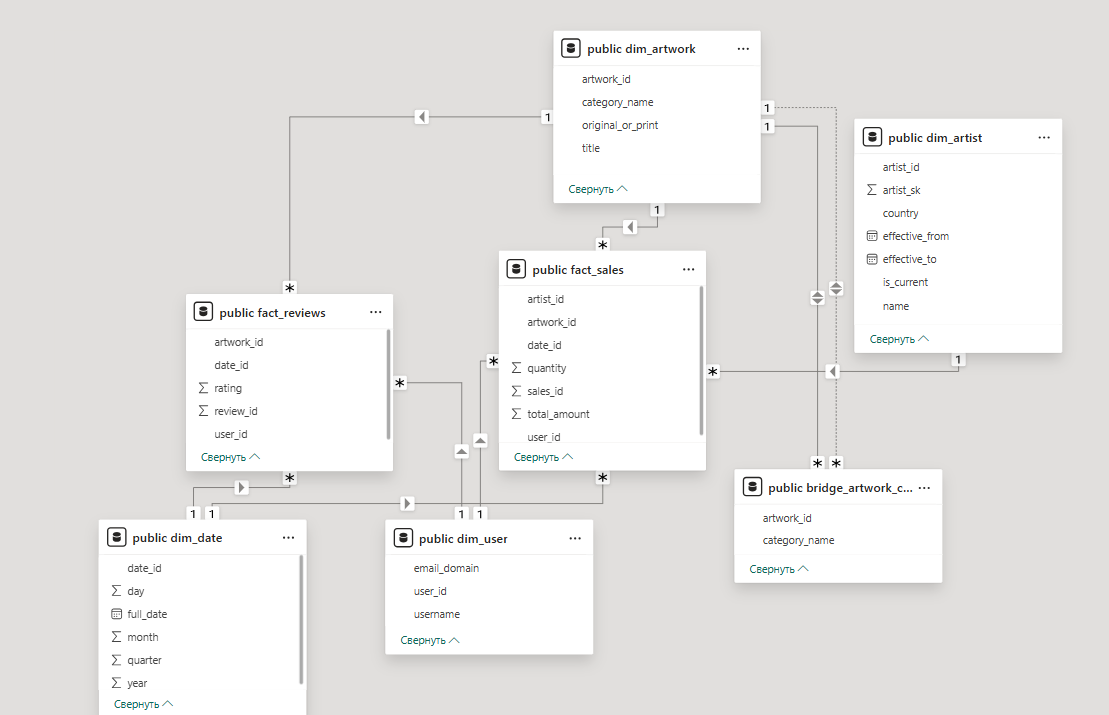
**Schema:**

****

### OLAP Schema:

* Implements a snowflake schema with dimension tables (dim\_date, dim\_artwork, dim\_artist, dim\_user) and fact tables (fact\_sales, fact\_reviews).
* A bridge table (bridge\_artwork\_category) supports a many-to-many relationship between artworks and categories.
* Uses surrogate keys such as artist\_sk in dim\_artist for decoupling from the operational artist\_id and supporting better version control.
* The table dim\_artist implements Slowly Changing Dimension Type 2 (SCD Type 2) to track historical changes in artist data using effective\_from, effective\_to, and is\_current fields.
* The table bridge\_artwork\_category handles many-to-many relationships between artworks and their categories, allowing aggregation by genre or style.
* Fact tables store aggregated data for analytical queries:
* fact\_sales includes quantity, total\_amount
* fact\_reviews stores rating values over time
* All tables are linked using foreign key constraints to ensure referential integrity and enable robust business intelligence reporting via tools like Power BI.

**Schema:**



**Part 2:** **Instructions how to run scripts:**(Run in order)

1. SCRIPT to create db tables of OLTP:

CREATE TABLE Users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

password\_hash VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Artists (

artist\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

bio TEXT,

birthdate DATE,

country VARCHAR(50)

);

CREATE TABLE Artworks (

artwork\_id SERIAL PRIMARY KEY,

artist\_id INT NOT NULL REFERENCES Artists(artist\_id),

title VARCHAR(150) NOT NULL,

description TEXT,

original\_or\_print VARCHAR(20) CHECK (original\_or\_print IN ('original', 'print')),

creation\_date DATE

);

CREATE TABLE Categories (

category\_id SERIAL PRIMARY KEY,

name VARCHAR(50) NOT NULL UNIQUE

);

CREATE TABLE Artwork\_Categories (

artwork\_id INT REFERENCES Artworks(artwork\_id),

category\_id INT REFERENCES Categories(category\_id),

PRIMARY KEY (artwork\_id, category\_id)

);

CREATE TABLE Inventory (

inventory\_id SERIAL PRIMARY KEY,

artwork\_id INT NOT NULL REFERENCES Artworks(artwork\_id),

quantity INT CHECK (quantity >= 0),

price NUMERIC(10,2) NOT NULL,

availability\_status VARCHAR(20) CHECK (availability\_status IN ('in\_stock', 'sold\_out', 'preorder'))

);

CREATE TABLE Orders (

order\_id SERIAL PRIMARY KEY,

user\_id INT NOT NULL REFERENCES Users(user\_id),

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status VARCHAR(20) CHECK (status IN ('pending', 'shipped', 'delivered', 'cancelled')) NOT NULL

);

CREATE TABLE Order\_Items (

order\_item\_id SERIAL PRIMARY KEY,

order\_id INT NOT NULL REFERENCES Orders(order\_id),

inventory\_id INT NOT NULL REFERENCES Inventory(inventory\_id),

quantity INT CHECK (quantity > 0),

unit\_price NUMERIC(10,2) NOT NULL

);

CREATE TABLE Payments (

payment\_id SERIAL PRIMARY KEY,

order\_id INT NOT NULL REFERENCES Orders(order\_id),

payment\_method VARCHAR(20) CHECK (payment\_method IN ('card', 'paypal', 'bank\_transfer')),

amount NUMERIC(10,2) NOT NULL,

payment\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Reviews (

review\_id SERIAL PRIMARY KEY,

artwork\_id INT NOT NULL REFERENCES Artworks(artwork\_id),

user\_id INT NOT NULL REFERENCES Users(user\_id),

rating INT CHECK (rating BETWEEN 1 AND 5),

comment TEXT,

review\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. SCRIPT to put data into OLTP

COPY Users(username, email, password\_hash)

FROM 'D:/SQLCourseWork/users.csv' DELIMITER ',' CSV HEADER;

COPY Artists(name, bio, birthdate, country)

FROM 'D:/SQLCourseWork/artists.csv' DELIMITER ',' CSV HEADER;

COPY Artworks(artist\_id, title, description, original\_or\_print, creation\_date)

FROM 'D:/SQLCourseWork/artworks.csv' DELIMITER ',' CSV HEADER;

COPY Categories(name)

FROM 'D:/SQLCourseWork/categories.csv' DELIMITER ',' CSV HEADER;

COPY Artwork\_Categories(artwork\_id, category\_id)

FROM 'D:/SQLCourseWork/artwork\_categories.csv' DELIMITER ',' CSV HEADER;

COPY Inventory(artwork\_id, quantity, price, availability\_status)

FROM 'D:/SQLCourseWork/inventory.csv' DELIMITER ',' CSV HEADER;

COPY Orders(user\_id, order\_date, status)

FROM 'D:/SQLCourseWork/orders.csv' DELIMITER ',' CSV HEADER;

COPY Order\_Items(order\_id, inventory\_id, quantity, unit\_price)

FROM 'D:/SQLCourseWork/order\_items.csv' DELIMITER ',' CSV HEADER;

COPY Payments(order\_id, payment\_method, amount, payment\_date)

FROM 'D:/SQLCourseWork/payments.csv' DELIMITER ',' CSV HEADER;

COPY Reviews(artwork\_id, user\_id, rating, comment, review\_date)

FROM 'D:/SQLCourseWork/reviews.csv' DELIMITER ',' CSV HEADER;

1. SCRIPT to create OLAP:

CREATE TABLE Dim\_Date (

date\_id SERIAL PRIMARY KEY,

full\_date DATE NOT NULL,

day INT,

month INT,

quarter INT,

year INT

);

CREATE TABLE Dim\_Artwork (

artwork\_id INT PRIMARY KEY,

title VARCHAR(150),

original\_or\_print VARCHAR(20),

category\_name VARCHAR(50)

);

CREATE TABLE Dim\_Artist (

artist\_sk SERIAL PRIMARY KEY,

artist\_id INT NOT NULL,

name VARCHAR(100),

country VARCHAR(50),

effective\_from DATE,

effective\_to DATE,

is\_current BOOLEAN

);

CREATE TABLE Dim\_User (

user\_id INT PRIMARY KEY,

username VARCHAR(50),

email\_domain VARCHAR(50)

);

CREATE TABLE Bridge\_Artwork\_Category (

artwork\_id INT NOT NULL,

category\_name VARCHAR(50) NOT NULL,

PRIMARY KEY (artwork\_id, category\_name)

);

CREATE TABLE Fact\_Sales (

sales\_id SERIAL PRIMARY KEY,

date\_id INT NOT NULL REFERENCES Dim\_Date(date\_id),

artwork\_id INT NOT NULL REFERENCES Dim\_Artwork(artwork\_id),

user\_id INT NOT NULL REFERENCES Dim\_User(user\_id),

artist\_id INT NOT NULL REFERENCES Dim\_Artist(artist\_sk),

quantity INT,

total\_amount NUMERIC(10, 2)

);

CREATE TABLE Fact\_Reviews (

review\_id INT PRIMARY KEY,

date\_id INT NOT NULL REFERENCES Dim\_Date(date\_id),

artwork\_id INT NOT NULL REFERENCES Dim\_Artwork(artwork\_id),

user\_id INT NOT NULL REFERENCES Dim\_User(user\_id),

rating INT CHECK (rating BETWEEN 1 AND 5)

);

4. SCRIPT to move data from OLTP database to OLAP database ( run in order #) ) :  
  
 SELECT dblink\_connect(

'oltp\_conn',

'dbname=CourseProject user=postgres password=тайна'

);

INSERT INTO Dim\_Date (full\_date, day, month, quarter, year)

SELECT DISTINCT date::DATE,

EXTRACT(DAY FROM date),

EXTRACT(MONTH FROM date),

EXTRACT(QUARTER FROM date),

EXTRACT(YEAR FROM date)

FROM (

SELECT order\_date::DATE AS date

FROM dblink('oltp\_conn', 'SELECT order\_date FROM Orders')

AS t(order\_date DATE)

UNION

SELECT payment\_date::DATE

FROM dblink('oltp\_conn', 'SELECT payment\_date FROM Payments')

AS t(payment\_date DATE)

UNION

SELECT review\_date::DATE

FROM dblink('oltp\_conn', 'SELECT review\_date FROM Reviews')

AS t(review\_date DATE)

) AS all\_dates

WHERE NOT EXISTS (

SELECT 1 FROM Dim\_Date d WHERE d.full\_date = all\_dates.date

);

INSERT INTO Dim\_User (user\_id, username, email\_domain)

SELECT user\_id, username, SPLIT\_PART(email, '@', 2)

FROM dblink('oltp\_conn', 'SELECT user\_id, username, email FROM Users')

AS t(user\_id INT, username TEXT, email TEXT)

WHERE NOT EXISTS (

SELECT 1 FROM Dim\_User du WHERE du.user\_id = t.user\_id

);

INSERT INTO Dim\_Artist (artist\_id, name, country, effective\_from, effective\_to, is\_current)

SELECT artist\_id, name, country, CURRENT\_DATE, NULL, TRUE

FROM dblink('oltp\_conn', 'SELECT artist\_id, name, country FROM Artists')

AS t(artist\_id INT, name TEXT, country TEXT)

WHERE NOT EXISTS (

SELECT 1 FROM Dim\_Artist da WHERE da.artist\_id = t.artist\_id AND da.is\_current = TRUE

);

INSERT INTO Dim\_Artwork (artwork\_id, title, original\_or\_print, category\_name)

SELECT a.artwork\_id, a.title, a.original\_or\_print, c.name

FROM dblink('oltp\_conn', 'SELECT artwork\_id, title, original\_or\_print FROM Artworks')

AS a(artwork\_id INT, title TEXT, original\_or\_print TEXT)

JOIN dblink('oltp\_conn', 'SELECT artwork\_id, category\_id FROM Artwork\_Categories')

AS ac(artwork\_id INT, category\_id INT) ON a.artwork\_id = ac.artwork\_id

JOIN dblink('oltp\_conn', 'SELECT category\_id, name FROM Categories')

AS c(category\_id INT, name TEXT) ON ac.category\_id = c.category\_id

WHERE NOT EXISTS (

SELECT 1 FROM Dim\_Artwork da WHERE da.artwork\_id = a.artwork\_id AND da.category\_name = c.name

);

INSERT INTO Bridge\_Artwork\_Category (artwork\_id, category\_name)

SELECT a.artwork\_id, c.name

FROM dblink('oltp\_conn', 'SELECT artwork\_id, category\_id FROM Artwork\_Categories')

AS a(artwork\_id INT, category\_id INT)

JOIN dblink('oltp\_conn', 'SELECT category\_id, name FROM Categories')

AS c(category\_id INT, name TEXT) ON a.category\_id = c.category\_id

WHERE NOT EXISTS (

SELECT 1 FROM Bridge\_Artwork\_Category b WHERE b.artwork\_id = a.artwork\_id AND b.category\_name = c.name

);

INSERT INTO Fact\_Sales (date\_id, artwork\_id, user\_id, artist\_id, quantity, total\_amount)

SELECT

d.date\_id,

i.artwork\_id,

o.user\_id,

da.artist\_sk,

oi.quantity,

(oi.quantity \* oi.unit\_price)

FROM dblink('oltp\_conn', 'SELECT order\_item\_id, order\_id, inventory\_id, quantity, unit\_price FROM Order\_Items')

AS oi(order\_item\_id INT, order\_id INT, inventory\_id INT, quantity INT, unit\_price NUMERIC)

JOIN dblink('oltp\_conn', 'SELECT order\_id, user\_id, order\_date FROM Orders')

AS o(order\_id INT, user\_id INT, order\_date DATE) ON oi.order\_id = o.order\_id

JOIN dblink('oltp\_conn', 'SELECT inventory\_id, artwork\_id FROM Inventory')

AS i(inventory\_id INT, artwork\_id INT) ON oi.inventory\_id = i.inventory\_id

JOIN dblink('oltp\_conn', 'SELECT artwork\_id, artist\_id FROM Artworks')

AS a(artwork\_id INT, artist\_id INT) ON i.artwork\_id = a.artwork\_id

JOIN Dim\_Date d ON d.full\_date = o.order\_date

JOIN Dim\_Artist da ON da.artist\_id = a.artist\_id AND da.is\_current = TRUE

WHERE NOT EXISTS (

SELECT 1 FROM Fact\_Sales fs

WHERE fs.date\_id = d.date\_id AND fs.artwork\_id = i.artwork\_id AND fs.user\_id = o.user\_id

);

INSERT INTO Fact\_Reviews (review\_id, date\_id, artwork\_id, user\_id, rating)

SELECT

r.review\_id,

d.date\_id,

r.artwork\_id,

r.user\_id,

r.rating

FROM dblink('oltp\_conn', 'SELECT review\_id, artwork\_id, user\_id, rating, review\_date FROM Reviews')

AS r(review\_id INT, artwork\_id INT, user\_id INT, rating INT, review\_date DATE)

JOIN Dim\_Date d ON d.full\_date = r.review\_date

WHERE NOT EXISTS (

SELECT 1 FROM Fact\_Reviews fr WHERE fr.review\_id = r.review\_id

);

**Part 3: Power BI report**

Power BI report representing 3 visual components:  
1) Sum of total sales by date.  
2) Quantity of goods sold by product name.  
3) Total sales by brand.  
All visual components can be sorted by day and month

**Part 4: SCRIPT to run queries based on OLTP and OLAP DBs:**

* OLTP:

Number of artworks sold by each artist:

SELECT a.name AS artist\_name, COUNT(i.artwork\_id) AS artworks\_sold

FROM Artists a

JOIN Artworks ar ON a.artist\_id = ar.artist\_id

JOIN Inventory i ON ar.artwork\_id = i.artwork\_id

JOIN Order\_Items oi ON i.inventory\_id = oi.inventory\_id

GROUP BY a.name

ORDER BY artworks\_sold DESC;

Average rating per artwork:

SELECT ar.title, AVG(r.rating) AS avg\_rating

FROM Artworks ar

JOIN Reviews r ON ar.artwork\_id = r.artwork\_id

GROUP BY ar.title

ORDER BY avg\_rating DESC;

Number of orders by status:  
SELECT status, COUNT(\*) AS total\_orders

FROM Orders

GROUP BY status;

* OLAP:  
  Total sales by year

SELECT dd.year, SUM(fs.total\_amount) AS total\_sales

FROM fact\_sales fs

JOIN dim\_date dd ON fs.date\_id = dd.date\_id

GROUP BY dd.year

ORDER BY dd.year;

Total sales by category:

SELECT bac.category\_name, SUM(fs.total\_amount) AS total\_sales

FROM fact\_sales fs

JOIN dim\_artwork da ON fs.artwork\_id = da.artwork\_id

JOIN bridge\_artwork\_category bac ON da.artwork\_id = bac.artwork\_id

GROUP BY bac.category\_name

ORDER BY total\_sales DESC;

Average rating by category:

SELECT bac.category\_name, AVG(fr.rating) AS avg\_rating

FROM fact\_reviews fr

JOIN dim\_artwork da ON fr.artwork\_id = da.artwork\_id

JOIN bridge\_artwork\_category bac ON da.artwork\_id = bac.artwork\_id

GROUP BY bac.category\_name

ORDER BY avg\_rating DESC;