Group 15 – INSY 661 Project Report

Team members:

Anqi Chen (261044081), Tehut Biru (261004817), Uzair Ahmad (261008635), Matthieu Jacquand (260728214), Mehdi Yachfine (261005628), Shehryar Hussain (260986031)

Section 1

Overview of Business Scenario

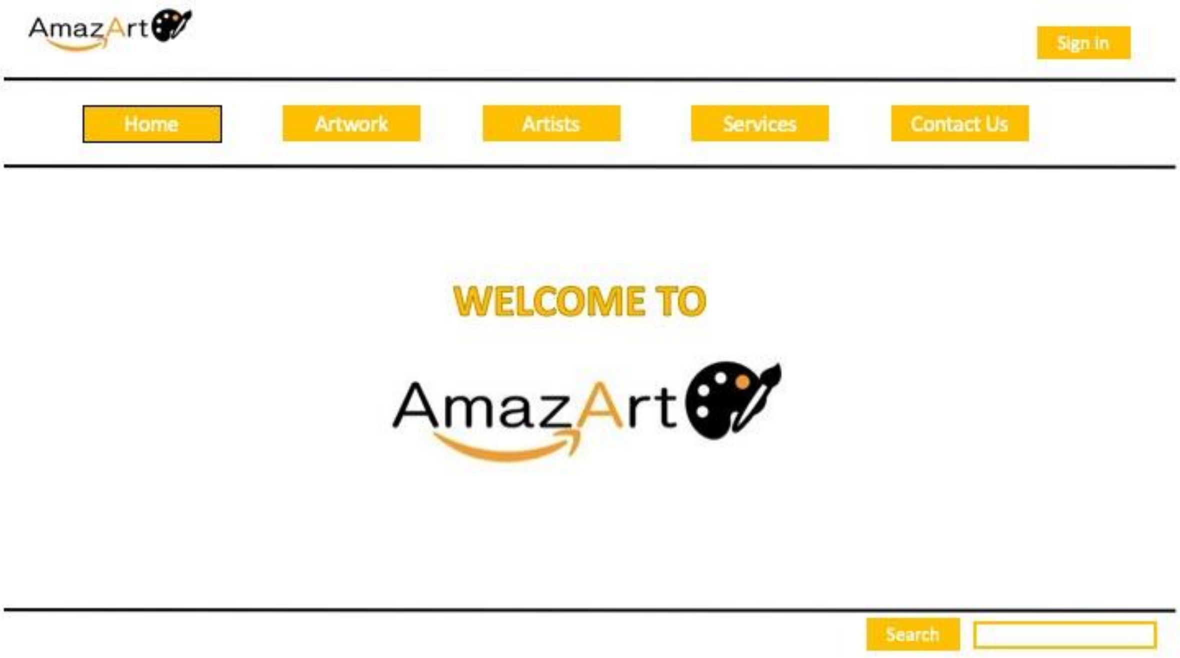
Description

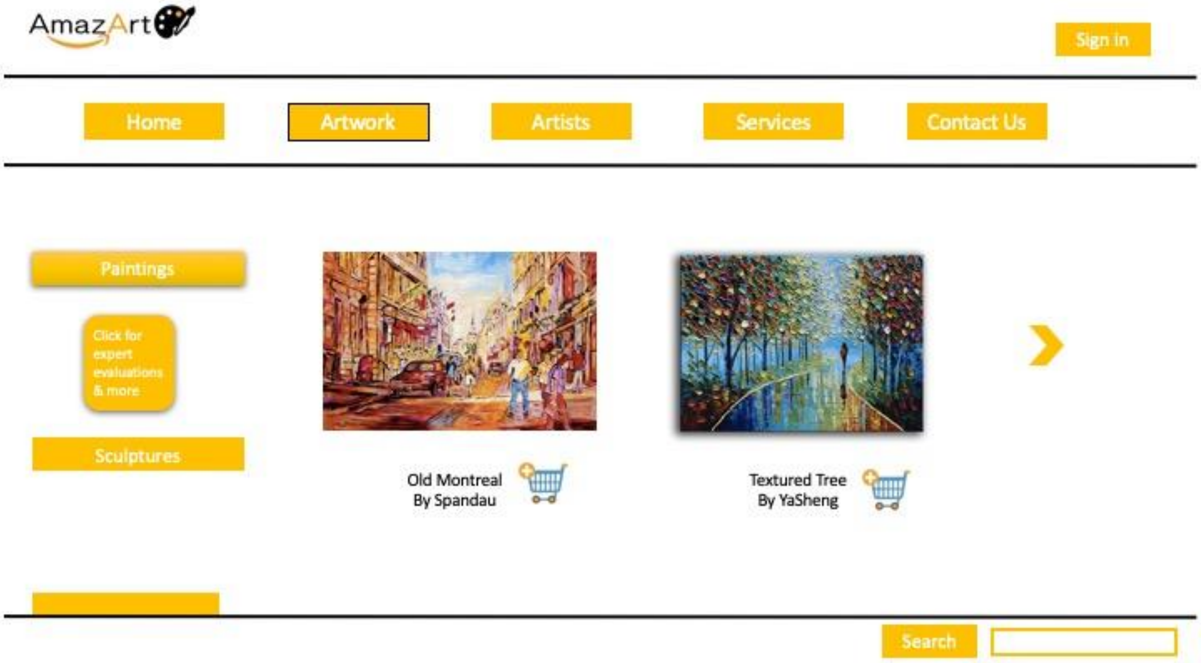
The past year has seen a massive growth in the value and volume of sales in digital art. NFT (Non-fungible token) marketplaces are booming and record sales are being recorded (See for reference: Beeple 69-million-dollar sale).

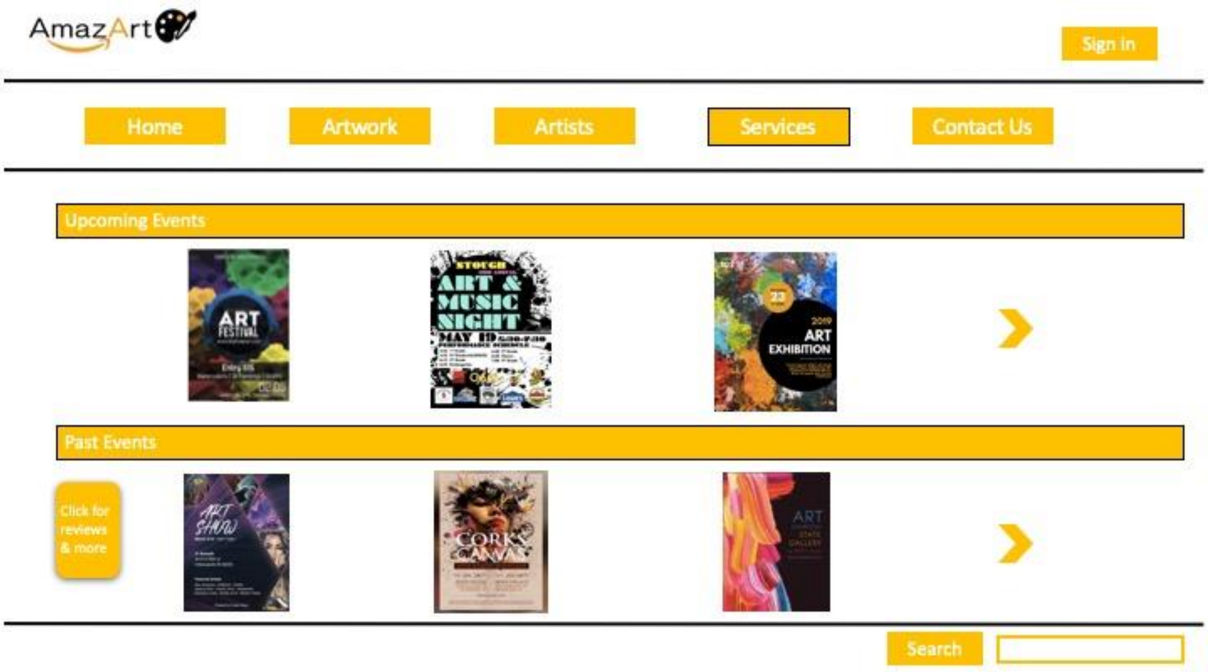
Overall, many small artists are finding recognition and the digital art market is at the highest it has ever been. **Amazon**, the world's biggest online retailer, has been keeping a close eye on the development of the digital art market and wants to step through the door. After brainstorming with the MMA consulting team, Amazon believes it can put forward a solution that can provide a robust infrastructure for art sales, providing **both traditional and digital art works.** This is where AmazArt comes in, which focuses on:

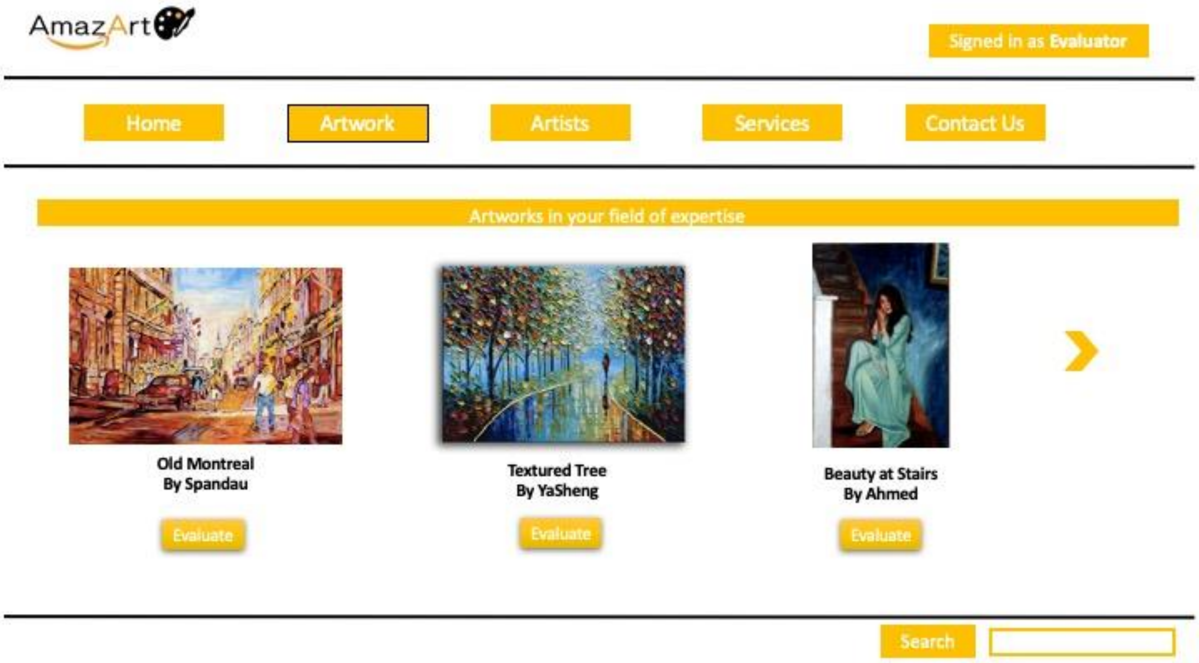
- Creating a platform centered around **user experience**, rallying passion and creating a sense of community. The key functionalities they want included are:
    - Provide access to a set of artists and with diverse artistic inclinations
    - Event management
    - Artwork evaluations
- They wish to provide a platform on which **both artists and buyers** can sign up to either sell or buy art.
    - NFT's and digital art
    - Traditional art (painting, poetry, drawing etc.)
        - Other related merchandise if desired
- They wish to provide payment options in **both traditional and crypto currencies.**
- They wish to provide **high service and reliability** in how the art is delivered from artist to client.
- They wish to provide a platform **for individuals, not companies**.
- They wish to provide **moderation and valuation** services to guarantee the quality of art sold.

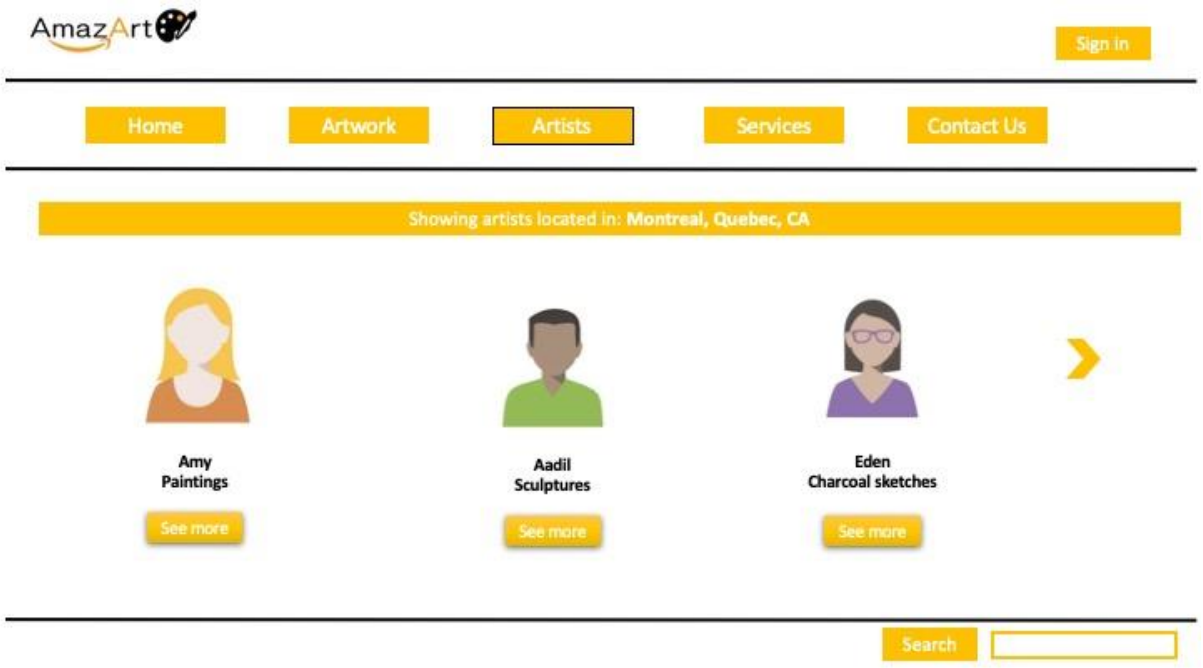
Website Demo: please find below prototypes for a few (sample) pages


AmazArt
Sign in
Home
Artwork
Artists
Services
Contact Us
WELCOME TO
AmazArt
Search


AmazArt
Sign in
Home
Artwork
Artists
Services
Contact Us
Paintings
Click for expert evaluations & more
Sculptures
Old Montreal By Spandau
Textured Tree By YaSheng
Search


AmazArt
Sign In
Home
Artwork
Artists
Services
Contact Us
Upcoming Events
ART FESTIVAL
ART & MUSIC NIGHT
2019 ART EXHIBITION
Past Events
Click for reviews & more
ART SHOW
CORKS CANVAS
ART
Search


AmazArt
Signed in as Evaluator
Home
Artwork
Artists
Services
Contact Us
Artworks in your field of expertise
Old Montreal
By Spandau
Evaluate
Textured Tree
By YaSheng
Evaluate
Beauty at Stairs
By Ahmed
Evaluate
Search



Showing artists located in: **Montreal, Quebec, CA**

Mission Statement

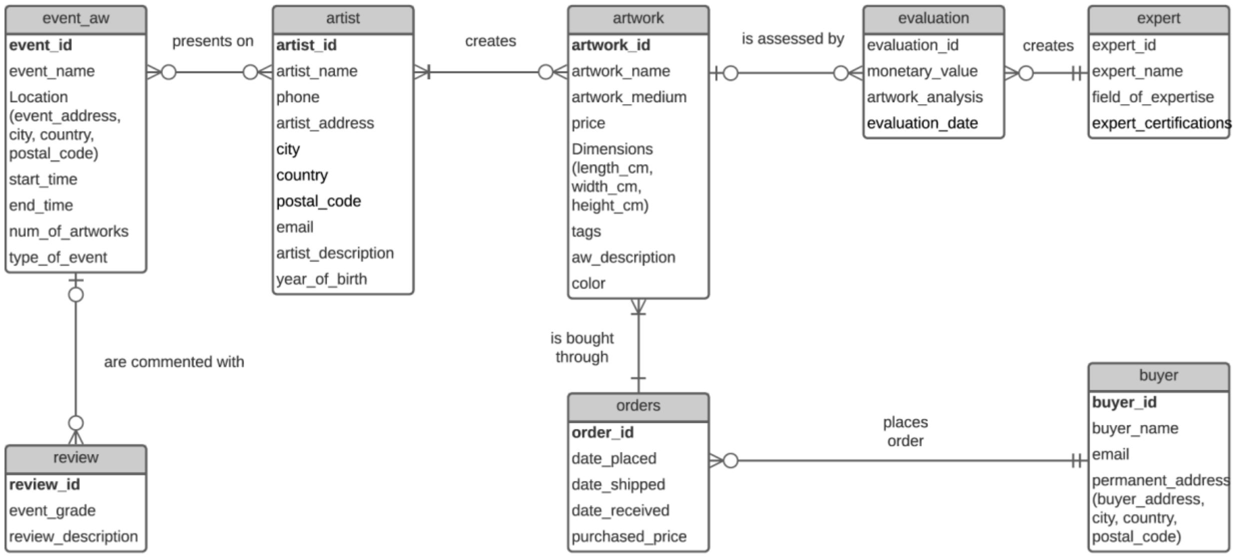
The purpose of the AmazArt database is to provide the infrastructure required to support a user-directed and secure marketplace for pieces by both traditional and digital artists. This infrastructure allows both the management of sales and orders, supports an intuitive and user-tailored interface and gives opportunities for its users to interact via different events.

Mission Objectives

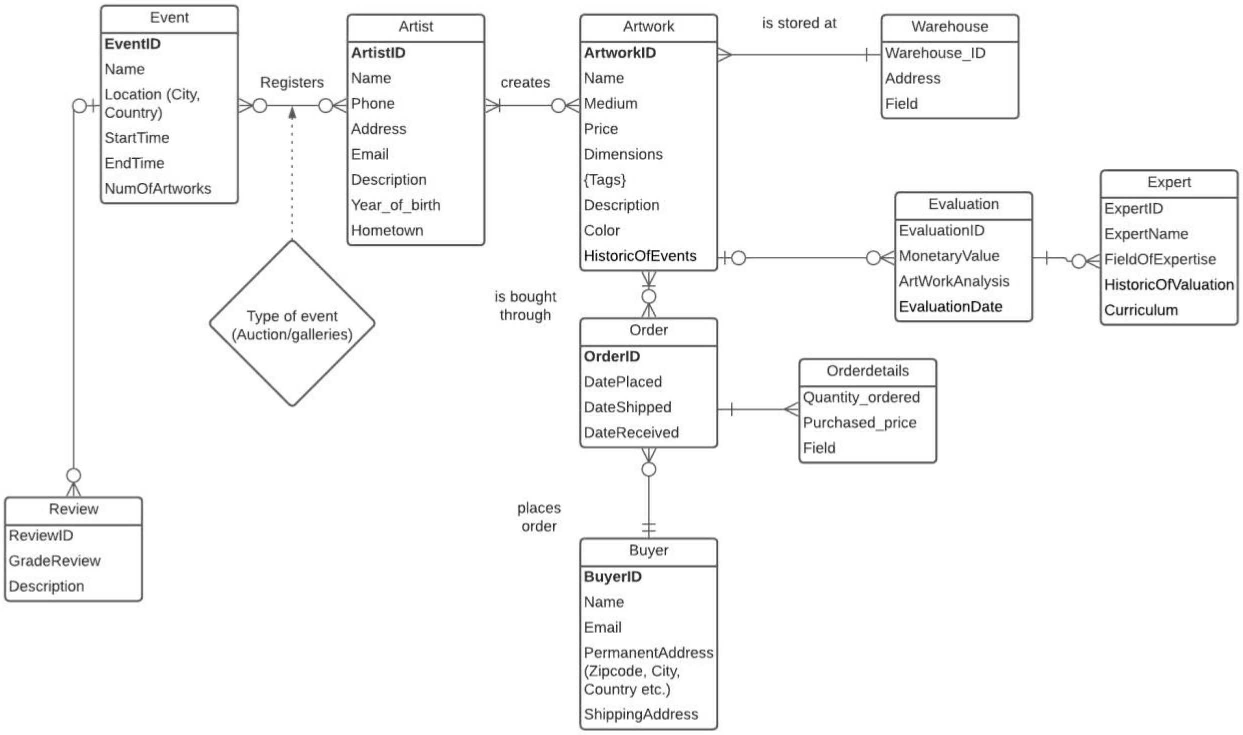
- **Data maintenance objectives**
  - To maintain (enter, update and delete) data on artists.
  - To maintain (enter, update and delete) data on buyers.
  - To maintain (enter, update and delete) data on experts.
  - To maintain (enter, update and delete) data on orders.
  - To maintain (enter, update and delete) data on artworks.
  - To maintain (enter, update and delete) data on tags.
  - To maintain (enter, update and delete) data on events.
  - To maintain (enter, update and delete) data on event reviews.
  - To maintain (enter, update and delete) data on artwork evaluations.
- **User services objectives**
  - To support user exploration of various artworks
  - To support event management and feedback.
  - To support reliable artwork purchases.
  - To support user evaluations in their field of expertise.
- **Dynamic hierarchical relationship**
  - To order and rank artists.
  - To order and rank artworks.
  - To order and rank events.

  - To order and rank interests.
  - To order and rank curators.
- **Generating reports, logs, summaries**
  - To generate and store logs on user data.
  - To generate and store logs on orders data.
  - To generate and store logs on review data.
  - To generate and store logs on evaluation data.
  - To generate and store logs on events data.

ERD



Alternative ERD considered



Data Dictionary

Description of Entities

Entity Name	Description	Aliases	Occurrence
event_aw	An entity that represents the events where artists' artworks are exhibited, as well as related information such as time, location, etc.	N/A	One artist can optionally present at multiple events, and one event can optionally be registered by multiple artists. One event can have multiple reviews, and one review can only be assigned to one event.
review	An entity that stores the reviews of events made by attendees. It includes review	N/A	One review can optionally be assigned to only one event, and one event

	description and grade.		can optionally have multiple reviews.
artist	An entity that shows the personal information of artists as well as the type of art they are invoved in	N/A	One artist can present at multiple events, and one event can have multiple artists to be presented at. One artist can create multiple artworks, while one artwork can be co-created by multiple artists.
artwork	An entity that has information about each piece of artwork, including its price, dimension, etc.	N/A	One artwork can be created by multiple artists, and one artist can create multiple artworks. One artwork can only be purchased by one order, each order can include multiple artworks. One artwork can have multiple evaluations, and one evaluation belongs to only one artwork.
orders	An entity that records each order placed, including the date and prices.	N/A	One order can include one or many artworks, one artwork can only be purchased in one order. One order belongs to only one buyer, and one buyer can place multiple orders.
evaluation	An entity that represents the evaluations of artworks made by experts.	N/A	One evaluation can optionally be made to only one artwork, and one artwork can optionally have multiple evaluations.

			One evaluation is created by one and only one expert, and one expert can optionally create multiple evaluations.
expert	An entity that records experts who give evaluations on artworks.	N/A	One expert can optionally create multiple evaluations, and one evaluation can be created by one and only one expert.
buyer	An entity that has all the information of buyers of artworks. It includes their contact information and their addresses for delivery.	N/A	One buyer can optionally place multiple orders, and one order can only be placed by one and only one buyer.
event_artist	An entity that connects event ID and artist ID, which have a many to many relationship tables.	N/A	It has one to many relationships to both artist table and event_aw table, as artist and event_aw have many to many relationships.
artist_artwork	An entity that connects artist ID and artwork ID, which have a many to many relationship tables.	N/A	It has one to many relationships to both artist and artwork tables, as artist and artwork have many to many relationships.

Description of Attributes

Entity Name	Attributes	Description	Data Type	Nulls	Multi-valued	Derived	Default
event_aw	Event_ID	Event unique identifier	VARCHAR(6)	No	No	No	None
	Event_Name	Name of the event	TEXT	Yes	No	No	None

	Event_address	The address where the event occurred	TEXT	Yes	No	No	None
	City	The city where the event occurred	TEXT	Yes	No	No	None
	Country	The country where the event occurred	TEXT	Yes	No	No	None
	Postal Code	The postal code where the event occurred	TEXT	Yes	No	No	None
	Start_Time	Time event starts	TIMESTAMP	Yes	No	No	None
	End _Time	Time event ends	TIMESTAMP	Yes	No	No	None
	Number_Of_Artworks	Number of artworks displayed at the event	INT	Yes	No	No	None
	Type_of_event	The kind of artistic event (e.g., auction, exhibition etc.)	TEXT	Yes	No	No	None
review	Review_ID	Review unique identifier	VARCHAR(6)	No	No	No	None
	Event_Grade	Review rating/grade given to event	VARCHAR(1)	Yes	No	No	None
	Review_Description	Written review given to event	TEXT	Yes	No	No	None
	Event_ID	The unique identifier connecting reviews with the event being reviewed	VARCHAR(6)	Yes	No	No	None
artist	Artist_ID	Artist’s unique identifier	VARCHAR(6)	No	No	No	None
	Artist_Name	Artist’s full name	TEXT	Yes	No	No	None
	Phone	Artist’s phone number	TEXT	Yes	No	No	None

	Artist_Address	Artist’s address	TEXT	Yes	No	No	None
	City	City where artist is located	TEXT	Yes	No	No	None
	Country	Country where artist is located	TEXT	Yes	No	No	None
	Postal_Code	Artist’s postal code	TEXT	Yes	No	No	None
	Email	Artist’s email address	TEXT	Yes	No	No	None
	Artist_Description	Description of the type of Art the Artist does	TEXT	Yes	No	No	None
	Year_of_Birth	Artist’s date of birth (year)	INT	Yes	No	No	None
artwork	Artwork_ID	Artwork unique identifier	VARCHAR(6)	No	No	No	None
	Artwork_Name	Title of Artwork	TEXT	Yes	No	No	None
	Artwork_Medium	The medium of artistic expression	TEXT	Yes	No	No	None
	Price	Price of the artwork	FLOAT	Yes	No	No	None
	Length_cm	Length of the artwork	FLOAT	Yes	No	No	None
	Width_cm	Width of the artwork	FLOAT	Yes	No	No	No
	Height_cm	Height of the artwork	FLOAT	Yes	No	No	No

	Tags	The art genre tags associated with the artwork	TEXT	Yes	No	No	None
	Aw_Description	Description of the artwork	TEXT	Yes	No	No	None
	Color	Main color contained in the artwork	TEXT	Yes	No	No	None
	Order_ID	The unique identifier connecting artworks to the orders entity	VARCHAR(6)	Yes	No	No	None
orders	Order_ID	Order unique identifier	VARCHAR(6)	No	No	No	None
	Date_Placed	Date order was placed	DATE	Yes	No	No	None
	Date_Shipped	Date order was shipped	DATE	Yes	No	No	None
	Date_Received	Date order was received by buyer	DATE	Yes	No	No	None
	Purchased_price	Price of the order	FLOAT	Yes	No	No	None
	Buyer_ID	The unique identifier connecting the order with the buyer entity	VARCHAR(6)	Yes	No	No	None
evaluation	Evaluation_ID	Evaluation unique identifier	VARCHAR(6)	No	No	No	None
	Monetary_Value	The expert price attached to that artwork	FLOAT	Yes	No	No	None
	Artwork_Analysis	Descriptive analysis of artwork	TEXT	Yes	No	No	None
	Evaluation_Date	The date the evaluation was made	DATE	Yes	No	No	None
	Expert_ID	The unique identifier connecting the	VARCHAR(6)	Yes	No	No	None

		evaluation with the expert entity					
	Artwork_ID	The unique identifier connecting the evaluation with the artwork entity	VARCHAR(6)	Yes	No	No	None
expert	Expert_ID	Expert unique identifier	VARCHAR(6)	No	No	No	None
	Expert_Name	Expert full name	TEXT	Yes	No	No	None
	Field_of_Expertise	The expert's field of expertise in Artwork	TEXT	Yes	No	No	None
	Expert_Certifications	Certifications related to artwork	TEXT	Yes	No	No	None
buyer	Buyer_ID	Buyer unique identifier	VARCHAR(6)	No	No	No	None
	Buyer_Name	Buyer full name	TEXT	Yes	No	No	None
	Email	Buyer email address	TEXT	Yes	No	No	None
	Buyer_Address	Buyer's address	TEXT	Yes	No	No	None
	City	City where buyer is located	TEXT	Yes	No	No	None
	Country	Country where buyer is located	TEXT	Yes	No	No	None
	Postal_Code	Buyer's postal code	TEXT	Yes	No	No	None
event_artist	Event_ID	Event unique identifier	VARCHAR(6)	No	No	No	None
	Artist_ID	Artist unique identifier	VARCHAR(6)	No	No	No	None
artist_artwork	Artist_ID	Artist unique identifier	VARCHAR(6)	No	No	No	None

	Artwork_ID	Artwork unique identifier	VARCHAR(6)	No	No	No	None

Relational Schema

PrimaryKey ForeignKey

event_aw (event_ID, event_Name, event_address, city, country, postal_code, start_time, end_time, num_of_artworks, type_of_event)

PK: event_id

review (review_id, event_grade, review_description, event_id)

PK: review_id

FK: event_id references event_aw (event_id)

artist (artist_id, artist_name, phone, artist_address, city, country, postal_code, email, artist_description, year_of_birth)

PK: artist_id

event_Artist (event_id, artist_id)

PK: event_id, artist_id

FK: event_id references event_aw (event_id)

artist_id references artist (artist_id)

artwork (artwork_id, artwork_name, artwork_medium, price, length_cm, width_cm, height_cm, tags, aw_description, color, order_id)

PK: artwork_id

FK: order_id references order (order_id)

artist_artwork (artist_id, artwork_id)

PK: artist_id, artwork_id

FK: artist_id references artist (artist_d)

Artwork_id references artwork (artwork_id)

orders (order_id, date_placed, date_shipped, date_received, purchased_price, buyer_id)

PK: order_id

FK: buyer_id references buyer (buyer_id)

evaluation (evaluation_id, monetary_value, artwork_analysis, evaluation_date, artwork_id, expert_id)

PK: evaluation_id

FK: artwork_id references artwork (artwork_id)

Expert_id references expert (expert_id)

expert (expert_id, expert_name, field_of_expertise, expert_certifications)

PK: expert_id

buyer (buyer_id, buyer_name, email, buyer_address, city, country, postal_code)

PK: buyer_id

Section 2

Accessing Database Remotely

The database is hosted on AWS. We made the following users:

1. Master_user (All priveleges including GRANT)
2. DBA_user (All rights on the schema)
3. Visitor (SELECT, SHOW VIEW, CALL PROCEDURE rights)

The credentials for the visitor have been shared for accessing the database remotely (please copy the text carefully):

Host: database-1.cu9lnwhoyxjz.us-east-2.rds.amazonaws.com

Port: 3306

User: visitor

Password: password

Views & Stored Procedure

The above credentials can be used to access the data in tables/views & also call procedures. Please execute the following line of code to get the results for all the 20 queries.

```
CALL groupproj.`20_queries`();
```

Following Views have also been created.

A. **Event_ratings (events with their review information)**

B. Artist_view (artworks details for an artist)
C. Buyer_view (Get details of all the artworks for a buyer)

Replicating the database

The following file was exported from MySQL Data Export wizard (please find the file attached separately).

Group15_Complete_
Database_Import_File

Please note that the views & stored procedures are included in the sql import file.

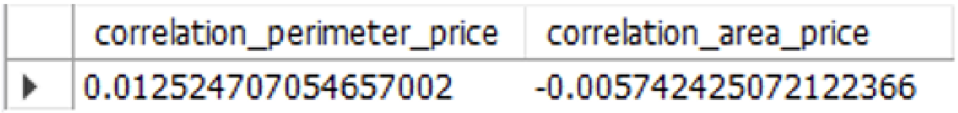
Queries

Query #1: Are larger artworks more expensive? Check if the price of an artwork is correlated with its perimeter or area.

Code:

```
WITH cte_corr AS
(SELECT  (count(*) * sum(x * y) - sum(x) * sum(y)) /
      (sqrt(count(*) * sum(x * x) - sum(x) * sum(x)) * sqrt(count(*) * sum(y * y) - sum(y) *
sum(y)))
      AS correlation_perimeter_price,
      (count(*) * sum(z * y) - sum(z) * sum(y)) /
      (sqrt(count(*) * sum(z * z) - sum(z) * sum(z)) * sqrt(count(*) * sum(y * y) - sum(y) * sum(y)))
      AS correlation_area_price
   FROM (SELECT price as y, length_cm + height_cm + width_cm as x, length_cm * height_cm *
width_cm z FROM artwork) derived)
SELECT * FROM cte_corr;
```

Output:



	correlation_perimeter_price	correlation_area_price
▶	0.012524707054657002	-0.005742425072122366

Query #2: What is the variation between expert evaluations for the artworks with artwork id 462, 95, 156, and 218?

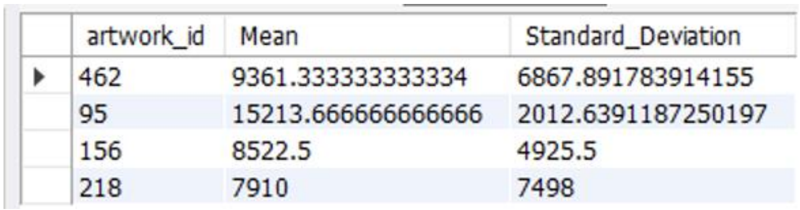
Code:

```
DROP TEMPORARY TABLE IF EXISTS temp_art;
CREATE TEMPORARY TABLE temp_art (artwork_id varchar(255));
```

```
INSERT INTO temp_art VALUES (462), (95), (156), (218);

SELECT artwork_id, AVG(monetary_value) 'Mean', STDDEV(monetary_value)
'Standard_Deviation' FROM evaluation e WHERE EXISTS
(SELECT * FROM temp_art WHERE e.artwork_id = artwork_id)
GROUP BY 1;
```

Output:



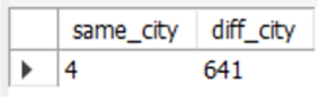
artwork_id	Mean	Standard_Deviation
462	9361.333333333334	6867.891783914155
95	15213.666666666666	2012.6391187250197
156	8522.5	4925.5
218	7910	7498

Query #3: Do buyers tend to buy artworks by artists who live in the same town as them or whose hometown is the same as their town of residence?

Code:

```
WITH cte_buyer AS
(SELECT buyer.buyer_id, city  as b_city, order_id FROM buyer join orders on buyer.buyer_id =
orders.buyer_id),
cte_artist AS
(SELECT artist.artist_id, city AS a_city , artwork.artwork_id, artwork.order_id FROM artist JOIN
artist_artwork JOIN artwork ON artist.artist_id = artist_artwork.artist_id AND
artwork.artwork_id = artist_artwork.artwork_id)
SELECT COUNT(CASE WHEN a_city=b_city THEN 1 ELSE NULL END) same_city, COUNT(CASE
WHEN a_city <> b_city THEN 1 ELSE NULL END) diff_city FROM cte_artist join cte_buyer on
cte_artist.order_id = cte_buyer.order_id;
```

Output:



same_city	diff_city
4	641

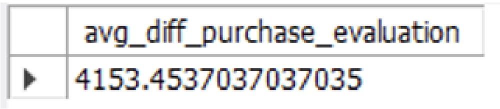
Query #4: What is the average difference between purchased price for an artwork and the monetary value assigned by the expert evaluator?

Code:

```
WITH cte_artwork AS
(SELECT artwork.artwork_id, AVG(purchased_price) 'purchased_price' FROM artwork JOIN
orders ON artwork.order_id = orders.order_id GROUP by 1),
cte_eval AS
```

```
(SELECT evaluation.artwork_id, AVG(monetary_value) monetary_value FROM evaluation GROUP
BY 1)
SELECT AVG((monetary_value - purchased_price)) avg_diff_purchase_evaluation FROM cte_eval
join cte_artwork ON cte_eval.artwork_id = cte_artwork.artwork_id;
```

Output:



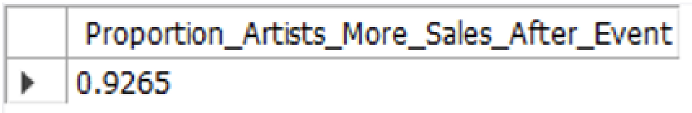
```
avg_diff_purchase_evaluation
4153.4537037037035
```

Query # 5: What proportion of artists saw an increase in sales after attending their first event (do not consider artists who had no sales before or after their first event)?

Code:

```
WITH cte_time AS
(SELECT event_artist.event_id, event_artist.artist_id, MIN(DATE(start_time)) event_day FROM
event_aw join event_artist on event_aw.event_id = event_artist.event_id GROUP BY 1, 2),
cte_artist AS
(SELECT artist_artwork.artist_id, date_placed FROM artist_artwork JOIN artwork JOIN orders ON
artwork.artwork_id = artist_artwork.artwork_id AND artwork.order_id = orders.order_id WHERE
artwork.order_id IS NOT NULL),
cte_sales AS
(SELECT cte_time.artist_id, COUNT(CASE WHEN date_placed >= event_day THEN 1 ELSE NULL
END) post_event, COUNT(CASE WHEN date_placed < event_day THEN 1 ELSE NULL END)
pre_event FROM cte_artist join cte_time on cte_time.artist_id = cte_artist.artist_id GROUP BY
1)
SELECT SUM(post_event > pre_event)/ COUNT(*) 'Proportion_Artists_More_Sales_After_Event'
FROM cte_sales;
```

Output:



```
Proportion_Artists_More_Sales_After_Event
0.9265
```

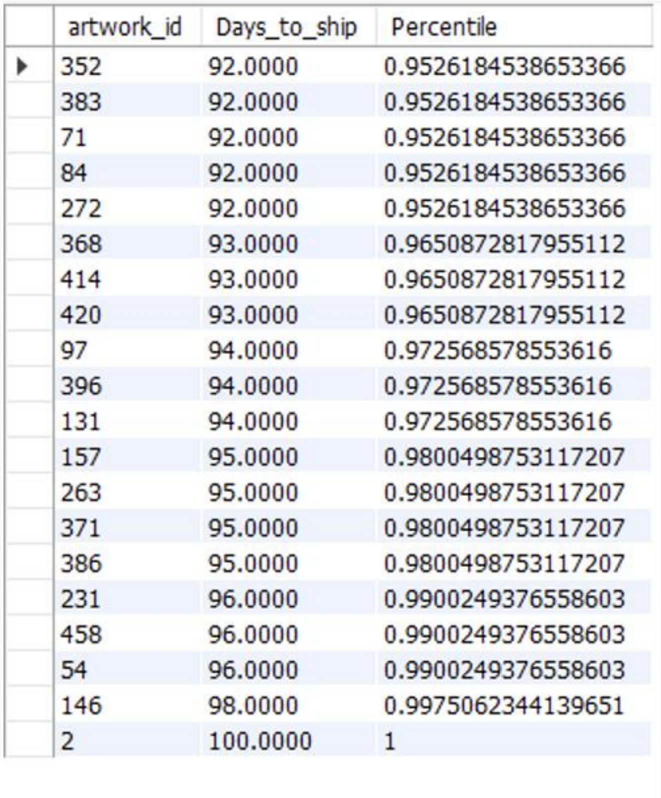
Query # 6: Get the top 5% of artworks that on average took the longest time to ship (date_received – date_placed). Order the 5% in ascending order.

Code:

```
WITH cte_ship AS
(SELECT artwork.artwork_id, AVG(DATEDIFF(date_received, date_placed)) 'Days_to_ship'  FROM
artwork JOIN orders ON artwork.order_id = orders.order_id WHERE artwork.order_id IS NOT
NULL GROUP BY 1),
percentiles AS
```

```
(SELECT artwork_id, Days_to_ship, PERCENT_RANK() OVER (ORDER BY Days_to_ship) AS
`Percentile` FROM cte_ship)
SELECT * FROM percentiles WHERE Percentile >= 0.95 ORDER BY Percentile ASC;
```

<u>Output:</u>



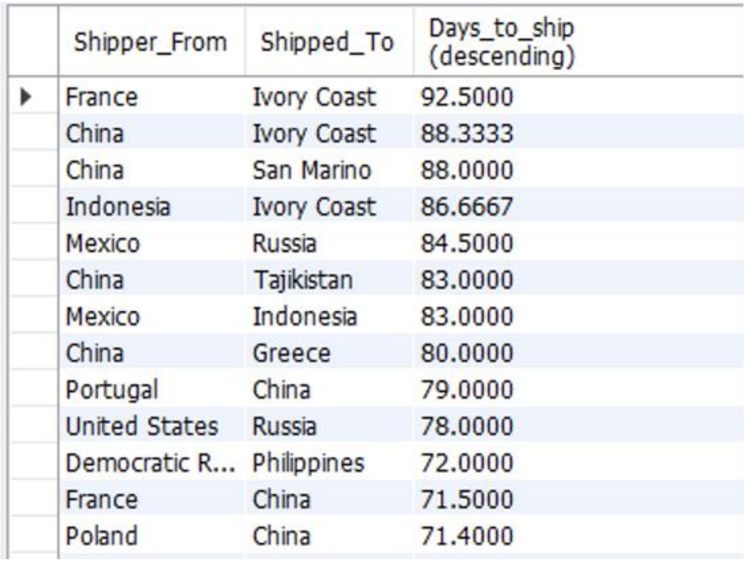
artwork_id	Days_to_ship	Percentile
352	92.0000	0.9526184538653366
383	92.0000	0.9526184538653366
71	92.0000	0.9526184538653366
84	92.0000	0.9526184538653366
272	92.0000	0.9526184538653366
368	93.0000	0.9650872817955112
414	93.0000	0.9650872817955112
420	93.0000	0.9650872817955112
97	94.0000	0.972568578553616
396	94.0000	0.972568578553616
131	94.0000	0.972568578553616
157	95.0000	0.9800498753117207
263	95.0000	0.9800498753117207
371	95.0000	0.9800498753117207
386	95.0000	0.9800498753117207
231	96.0000	0.9900249376558603
458	96.0000	0.9900249376558603
54	96.0000	0.9900249376558603
146	98.0000	0.9975062344139651
2	100.0000	1

<u>Query # 7:</u> Find the average time it takes to ship from an artist to a buyer considering their countries. What is the pair of artist's country and buyer's country, takes the longest time to ship (order by time descending). Consider records where at least 2 orders are placed.

<u>Code:</u>

```
WITH cte_buyer AS
(SELECT buyer.buyer_id, country  as country, order_id, date_placed, date_received FROM buyer
join orders on buyer.buyer_id = orders.buyer_id),
cte_artist AS
(SELECT artist.artist_id, country AS country, artwork.artwork_id, artwork.order_id FROM artist
JOIN artist_artwork JOIN artwork ON artist.artist_id = artist_artwork.artist_id AND
artwork.artwork_id = artist_artwork.artwork_id)
SELECT cte_artist.country 'Shipper_From', cte_buyer.country 'Shipped_To',
AVG(DATEDIFF(date_received, date_placed)) 'Days_to_ship (descending)' FROM cte_artist join
cte_buyer on cte_artist.order_id = cte_buyer.order_id GROUP BY 1, 2 HAVING COUNT(*) >= 2
ORDER BY 3 DESC;
```

Output:



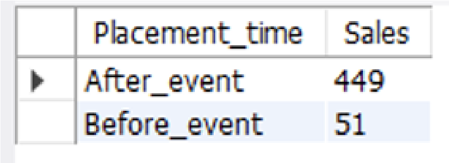
Shipper_From	Shipped_To	Days_to_ship (descending)
France	Ivory Coast	92.5000
China	Ivory Coast	88.3333
China	San Marino	88.0000
Indonesia	Ivory Coast	86.6667
Mexico	Russia	84.5000
China	Tajikistan	83.0000
Mexico	Indonesia	83.0000
China	Greece	80.0000
Portugal	China	79.0000
United States	Russia	78.0000
Democratic R...	Philippines	72.0000
France	China	71.5000
Poland	China	71.4000

Query # 8: In the week before and after the most popular event (most artists attended), do we see an increase in the overall orders placed?

Code:

```
WITH cte_time AS
(SELECT event_aw.event_id, DATE(event_aw.start_time) event_day, COUNT(*)
event_attendance FROM event_aw join event_artist on event_aw.event_id =
event_artist.event_id GROUP BY 1, 2),
cte_event AS
(SELECT min(event_day) AS 'date_popular_event' FROM cte_time WHERE event_attendance =
(SELECT MAX(event_attendance) FROM cte_time)),
cte_artist AS
(SELECT date_placed FROM artist_artwork JOIN artwork JOIN orders ON artwork.artwork_id =
artist_artwork.artwork_id AND artwork.order_id = orders.order_id WHERE artwork.order_id IS
NOT NULL)
SELECT CASE WHEN date_placed >= date_popular_event THEN "After_event" ELSE
"Before_event" END `Placement_time`, COUNT(*) 'Sales' FROM orders, cte_event GROUP BY 1;
```

Output:



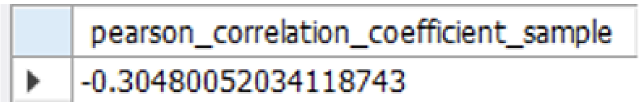
Placement_time	Sales
After_event	449
Before_event	51

Query # 9: Rank the artists based on their sales before and after a certain date `June 10th 2021` and find rank correlation.

Code:

```
SET @ref_date = '2021-06-10';
WITH cte_artist AS
(SELECT artist_artwork.artist_id, SUM(CASE WHEN date_placed >= @ref_date THEN 1 ELSE 0
END) 'After_date', SUM(CASE WHEN date_placed < @ref_date THEN 1 ELSE 0 END)
'Before_data', min(date_placed) FROM artist_artwork JOIN artwork JOIN orders ON
artwork.artwork_id = artist_artwork.artwork_id AND artwork.order_id = orders.order_id WHERE
artwork.order_id IS NOT NULL GROUP BY 1),
rank_order AS
(SELECT artist_id, DENSE_RANK() OVER (ORDER BY After_date) `Rank_After`, DENSE_RANK()
OVER (ORDER BY Before_data) `Rank_Before` FROM cte_artist),
cte_corr AS
( SELECT  (count(*) * sum(x * y) - sum(x) * sum(y)) /
      (sqrt(count(*) * sum(x * x) - sum(x) * sum(x)) * sqrt(count(*) * sum(y * y) - sum(y) *
sum(y)))
      AS pearson_correlation_coefficient_sample
   FROM (SELECT Rank_After as y, Rank_Before as x FROM rank_order) derived)
SELECT * FROM cte_corr;
```

Output:



```
pearson_correlation_coefficient_sample
-0.30480052034118743
```

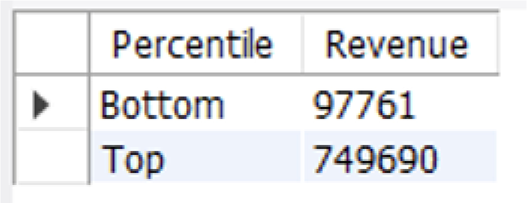
Query # 10: Compare the revenue generated by the most expensive artworks (top 20%) vs least expensive artworks (bottom 20%).

Code:

```
SET @pc_low = 0.2;
SET @pc_high = 0.8;

WITH cte_order
AS (SELECT orders.order_id, SUM(price) 'revenue' FROM artwork JOIN orders on
artwork.order_id = orders.order_id GROUP BY 1),
cte_rank
AS (SELECT order_id, revenue, PERCENT_RANK() OVER (ORDER BY revenue) AS `Percentile`
FROM cte_order)
SELECT CASE WHEN Percentile >= @pc_high THEN 'Top' WHEN Percentile <= @pc_low THEN
'Bottom' ELSE 'Other' END 'Percentile', SUM(revenue) 'Revenue' FROM cte_rank WHERE
Percentile NOT BETWEEN @pc_low and @pc_high GROUP BY 1;
```

Output:



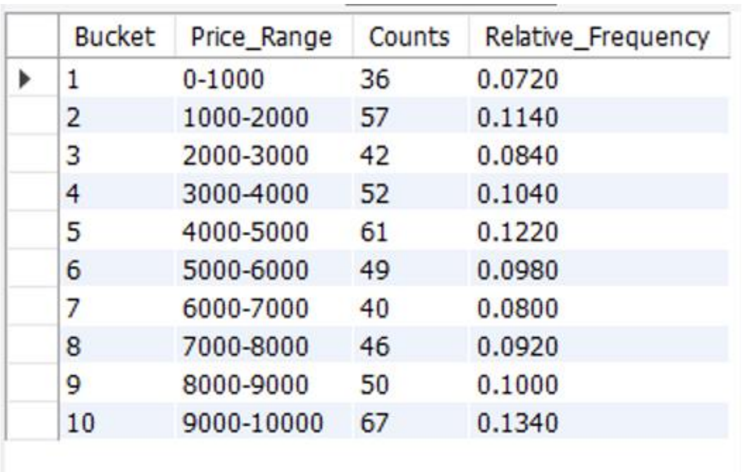
	Percentile	Revenue
▶	Bottom	97761
	Top	749690

Query # 11: Make 5 buckets/ranges of artworks based on price and find their relative frequencies.

Code:

```
WITH cte_counts AS
(SELECT CEIL(price/1000) 'Bucket', CONCAT(1000*FLOOR(price/1000), '-',
1000*CEILING(price/1000)) `Price_Range`, COUNT(*) `Counts` FROM artwork GROUP BY 1),
cte_max AS
(SELECT SUM(Counts) `Sum_counts` FROM cte_counts)
SELECT cte_counts.*, Counts/Sum_counts 'Relative_Frequency'  FROM cte_counts join cte_max
ORDER BY 1 ASC;
```

Output:



	Bucket	Price_Range	Counts	Relative_Frequency
▶	1	0-1000	36	0.0720
	2	1000-2000	57	0.1140
	3	2000-3000	42	0.0840
	4	3000-4000	52	0.1040
	5	4000-5000	61	0.1220
	6	5000-6000	49	0.0980
	7	6000-7000	40	0.0800
	8	7000-8000	46	0.0920
	9	8000-9000	50	0.1000
	10	9000-10000	67	0.1340

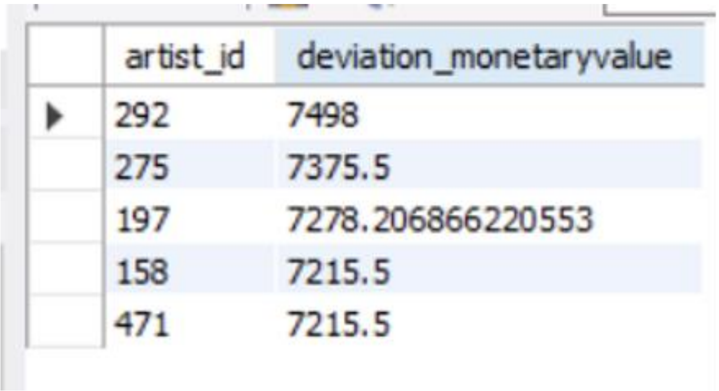
Query # 12: Find out the 5 most inconsistently evaluated artist. This should be the artist that has the most variation in monetary value by experts.

Code:

```
SELECT artist.artist_id, STDDEV(evaluation.monetary_value) as deviation_monetaryvalue FROM
artist
INNER JOIN artist_artwork ON artist.artist_id = artist_artwork.artist_id
INNER JOIN artwork ON artist_artwork.artwork_id = artwork.artwork_id
INNER JOIN evaluation ON evaluation.artwork_id = artwork.artwork_id
GROUP BY artist.artist_id
```

```
        ORDER BY  deviation_monetaryvalue DESC LIMIT 5;
```

Output:



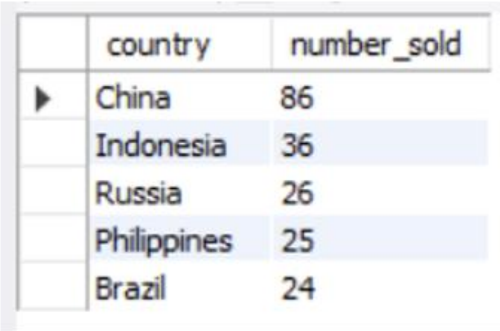
artist_id	deviation_monetaryvalue
292	7498
275	7375.5
197	7278.20686220553
158	7215.5
471	7215.5

Query # 13: Which are the top 5 art buying countries?

Code:

```
        SELECT buyer.country, COUNT(DISTINCT artwork.order_id) as number_sold FROM artwork
        INNER JOIN orders ON artwork.order_id = orders.order_id
        INNER JOIN buyer ON buyer.buyer_id = orders.buyer_id
        GROUP BY buyer.country
        ORDER BY number_sold DESC LIMIT 5;
```

Output:



country	number_sold
China	86
Indonesia	36
Russia	26
Philippines	25
Brazil	24

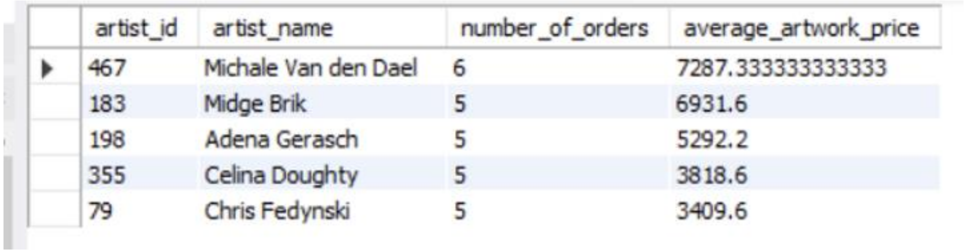
Query # 14: Whose artwork is the most popular ones? Assess the popularity based on the number of orders of artwork this artist creates. Show artist name, id, number of orders, average price of artwork of orders of this artist. Rank the average artwork price in descending order, and show only 5 records.

Code:

```
        SELECT artist_artwork.artist_id, artist.artist_name, COUNT(artwork.order_id) as
        number_of_orders,
        avg(artwork.price) as average_artwork_price
        FROM artwork
```

```
INNER JOIN artist_artwork ON artist_artwork.artwork_id = artwork.artwork_id
INNER JOIN artist ON artist.artist_id = artist_artwork.artist_id
GROUP BY artist_artwork.artist_id
ORDER BY number_of_orders DESC LIMIT 5;
```

Output:



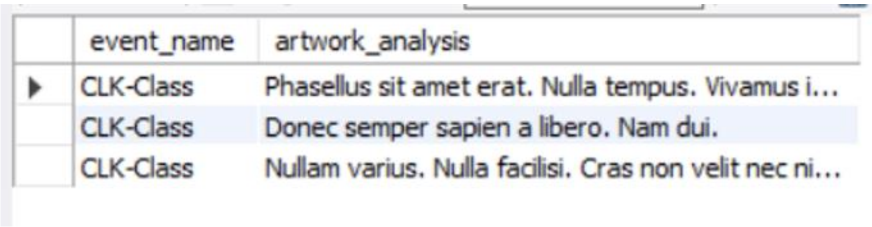
	artist_id	artist_name	number_of_orders	average_artwork_price
▶	467	Michale Van den Dael	6	7287.333333333333
	183	Midge Brik	5	6931.6
	198	Adena Gerasch	5	5292.2
	355	Celina Doughty	5	3818.6
	79	Chris Fedynski	5	3409.6

Query # 15: Show all the evaluations for artworks presented on event “CLK-Class”.

Code:

```
SET  @event_name = 'CLK-Class';
 SELECT event_aw.event_name, evaluation.artwork_analysis FROM event_aw
 INNER JOIN event_artist ON event_aw.event_id = event_artist.event_id
 INNER JOIN artist ON event_artist.artist_id = artist.artist_id
 INNER JOIN artist_artwork ON artist_artwork.artist_id = artist.artist_id
 INNER JOIN artwork ON artwork.artwork_id = artist_artwork.artwork_id
 INNER JOIN evaluation ON evaluation.artwork_id = artwork.artwork_id
 WHERE event_aw.event_name = @event_name;
```

Output:



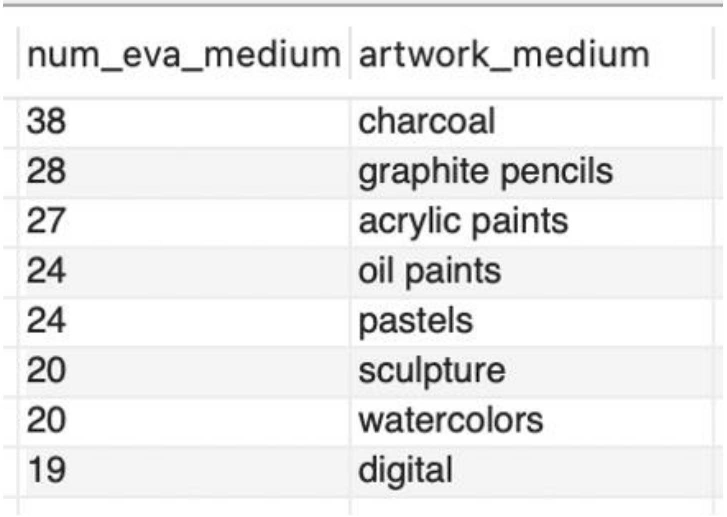
	event_name	artwork_analysis
▶	CLK-Class	Phasellus sit amet erat. Nulla tempus. Vivamus i...
	CLK-Class	Donec semper sapien a libero. Nam dui.
	CLK-Class	Nullam varius. Nulla facilisi. Cras non velit nec ni...

Query # 16: How many evaluations are given based on the different types of artworks medium? Order the number of evaluations in descending order.

Code:

```
SELECT COUNT(evaluation_id) AS num_eva_medium, artwork.artwork_medium
FROM artwork
JOIN evaluation ON artwork.artwork_id = evaluation.artwork_id
GROUP BY artwork.artwork_medium
ORDER BY num_eva_medium DESC;
```

Output:



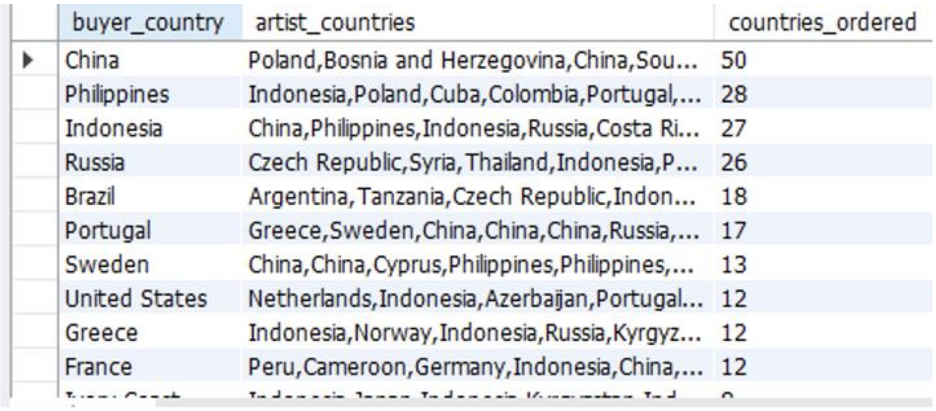
num_eva_medium	artwork_medium
38	charcoal
28	graphite pencils
27	acrylic paints
24	oil paints
24	pastels
20	sculpture
20	watercolors
19	digital

Query # 17: For a given buyer's country, find out all the countries they have bought art from. Which countries have bought art from the most diverse set of countries (sort descending)?

Code:

```
SELECT buyer.country buyer_country, GROUP_CONCAT(artist.country) AS artist_countries ,
COUNT(DISTINCT artist.country)  countries_ordered
FROM buyer
JOIN orders JOIN artwork JOIN artist_artwork JOIN artist  ON buyer.buyer_id = orders.buyer_id
AND artwork.order_id = orders.order_id AND artwork.artwork_id = artist_artwork.artwork_id
AND  artist.artist_id = artist_artwork.artist_id
GROUP BY buyer.country ORDER BY COUNT(DISTINCT artist.country) DESC;
```

Output:



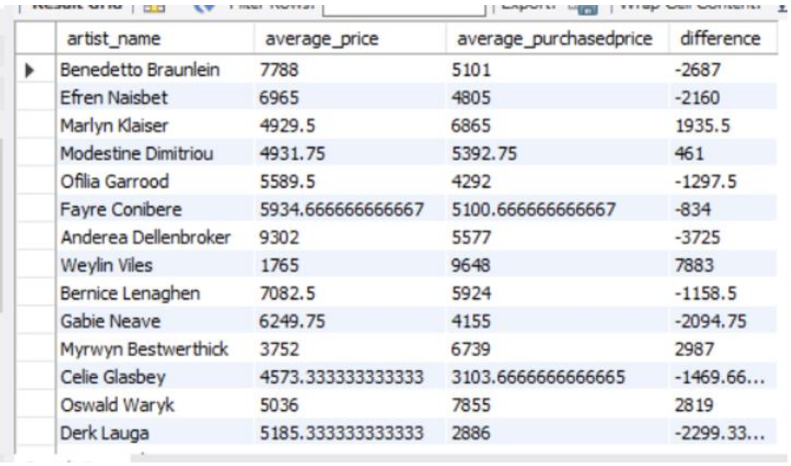
buyer_country	artist_countries	countries_ordered
China	Poland,Bosnia and Herzegovina,China,Sou...	50
Philippines	Indonesia,Poland,Cuba,Colombia,Portugal,...	28
Indonesia	China,Philippines,Indonesia,Russia,Costa Ri...	27
Russia	Czech Republic,Syria,Thailand,Indonesia,P...	26
Brazil	Argentina,Tanzania,Czech Republic,Indon...	18
Portugal	Greece,Sweden,China,China,China,Russia,...	17
Sweden	China,China,Cyprus,Philippines,Philippines,...	13
United States	Netherlands,Indonesia,Azerbaijan,Portugal...	12
Greece	Indonesia,Norway,Indonesia,Russia,Kyrgyz...	12
France	Peru,Cameroon,Germany,Indonesia,China,...	12
Ivory Coast	Indonesia,Japan,Indonesia,Kyrgyzstan,Ind...	9

Query # 18: Find the difference in average price and average purchased price of artworks of an artist.

Code:

```
SELECT artist.artist_name, avg(artwork.price) as average_price, avg(orders.purchased_price) as
average_purchasedprice, (avg(orders.purchased_price) - avg(artwork.price)) as difference
FROM artist
INNER JOIN artist_artwork ON artist.artist_id = artist_artwork.artist_id
INNER JOIN artwork ON artwork.artwork_id = artist_artwork.artwork_id
INNER JOIN orders ON orders.order_id = artwork.order_id
GROUP BY artist.artist_name;
```

Output:



artist_name	average_price	average_purchasedprice	difference
Benedetto Braunlein	7788	5101	-2687
Efren Naisbet	6965	4805	-2160
Marlyn Klaiser	4929.5	6865	1935.5
Modestine Dimitriou	4931.75	5392.75	461
Ofilia Garrood	5589.5	4292	-1297.5
Fayre Conibere	5934.666666666667	5100.666666666667	-834
Anderea Dellenbroker	9302	5577	-3725
Weylin Viles	1765	9648	7883
Bernice Lenaghen	7082.5	5924	-1158.5
Gabie Neave	6249.75	4155	-2094.75
Myrwyn Bestwerthick	3752	6739	2987
Celie Glasbey	4573.333333333333	3103.6666666666665	-1469.66...
Oswald Waryk	5036	7855	2819
Derk Lauga	5185.333333333333	2886	-2299.33...

Query # 19: What is the average time a buyer waits before making their next purchase? Compare dates of placing orders and find the average time between the orders. Show the results for buyers that have placed at least 2 orders.

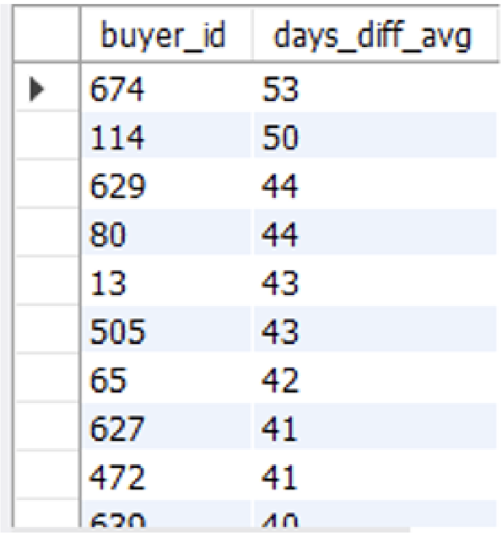
Code:

```
DROP TEMPORARY TABLE IF EXISTS dt;
CREATE TEMPORARY TABLE dt
SELECT * FROM orders order by buyer_id, date_placed;
DROP TEMPORARY TABLE IF EXISTS cte_row_temp_1;
CREATE TEMPORARY TABLE cte_row_temp_1
SELECT *, ROW_NUMBER() OVER(PARTITION BY buyer_id) AS row_num FROM dt ORDER BY
buyer_id ASC, date_placed ASC;
DROP TEMPORARY TABLE IF EXISTS cte_row_temp_2;
CREATE TEMPORARY TABLE cte_row_temp_2
SELECT * FROM cte_row_temp_1;

SELECT r1.buyer_id, DATEDIFF(r2.date_placed, r1.date_placed) days_diff_avg

FROM cte_row_temp_1 AS r1 join cte_row_temp_2 AS r2 ON r1.row_num+1 = r2.row_num
AND r1.buyer_id = r2.buyer_id GROUP BY 1 ORDER BY 2 DESC;
```

Output:



buyer_id	days_diff_avg
674	53
114	50
629	44
80	44
13	43
505	43
65	42
627	41
472	41
620	40

Query # 20: Are certain types of events assigning a higher review grade on average? To make it more complicated: add filter by location, is that type of event even more popular in a certain location?

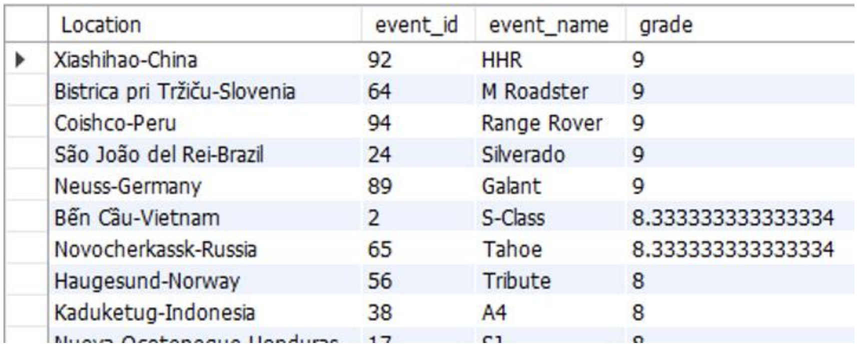
Code:

```
 -- Using View event_ratings
CREATE  VIEW `groupproj`.`event_ratings` AS
   SELECT
      `groupproj`.`event_aw`.`event_id` AS `event_id`,
      `groupproj`.`event_aw`.`event_name` AS `event_name`,
      `groupproj`.`event_aw`.`type_of_event` AS `type_of_event`,
      `groupproj`.`event_aw`.`event_address` AS `event_address`,
      `groupproj`.`event_aw`.`city` AS `city`,
      `groupproj`.`event_aw`.`country` AS `country`,
      `groupproj`.`event_aw`.`postal_code` AS `postal_code`,
      `groupproj`.`event_aw`.`start_time` AS `start_time`,
      `groupproj`.`event_aw`.`end_time` AS `end_time`,
      `groupproj`.`event_aw`.`number_of_artworks` AS `number_of_artworks`,
      `groupproj`.`review`.`review_id` AS `review_id`,
      `groupproj`.`review`.`event_grade` AS `event_grade`,
      `groupproj`.`review`.`review_description` AS `review_description`
   FROM
      (`groupproj`.`event_aw`
      LEFT JOIN `groupproj`.`review` ON ((`groupproj`.`event_aw`.`event_id` =
`groupproj`.`review`.`event_id`)))
   ORDER BY `groupproj`.`event_aw`.`event_id`;

WITH cte_eventgrade
```

```
AS(SELECT CONCAT(city, "-", country) Location, event_id, event_name, AVG(event_grade) grade
FROM event_ratings GROUP BY 1, 2, 3 ORDER BY grade DESC),
cte_location AS
(SELECT LOCATION, MAX(grade) max_grade FROM cte_eventgrade GROUP BY 1)
SELECT cte_eventgrade.* FROM cte_eventgrade join cte_location on cte_location.max_grade =
cte_eventgrade.grade and cte_location.location = cte_eventgrade.location;
```

Output:



Location	event_id	event_name	grade
Xiashihao-China	92	HHR	9
Bistrica pri Tržiču-Slovenia	64	M Roadster	9
Coishco-Peru	94	Range Rover	9
São João del Rei-Brazil	24	Silverado	9
Neuss-Germany	89	Galant	9
Bến Cầu-Vietnam	2	S-Class	8.333333333333334
Novocherkassk-Russia	65	Tahoe	8.333333333333334
Haugesund-Norway	56	Tribute	8
Kaduketug-Indonesia	38	A4	8
Nueva Ocotepeque-Honduras	17	S1	8

Section 3

Description of 1-2 interesting queries:

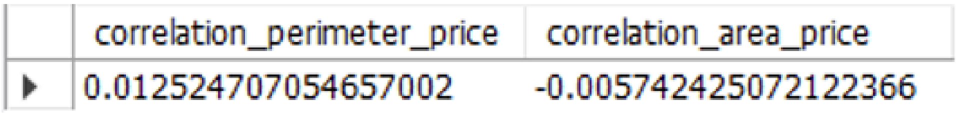
Query #1: Are larger artworks more expensive? Check if the price of an artwork is correlated with its perimeter or area.

A simple descriptive analysis could be to figure out if a particular variable is linearly correlated with another. We want to figure out if the size of the artwork determines its monetary value. For our analysis, we used the area & perimeter of an artwork and find the strength of their correlation with price.

Pearson's Correlation Coefficient is a linear correlation coefficient that returns a value of between -1 and +1. A -1 means there is a strong negative correlation and +1 means that there is a strong positive correlation. A 0 means that there is no correlation. We have simply used this formula in our query.

$$r=\frac{\sum\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sqrt{\sum\left(x_i-\bar{x}\right)^2\sum\left(y_i-\bar{y}\right)^2}}$$

Based on our analysis, there is a weak positive (linear) correlation between perimeter & price, and a very weak negative (linear) correlation between area of an artwork & price.



correlation_perimeter_price	correlation_area_price
0.012524707054657002	-0.005742425072122366

Code:

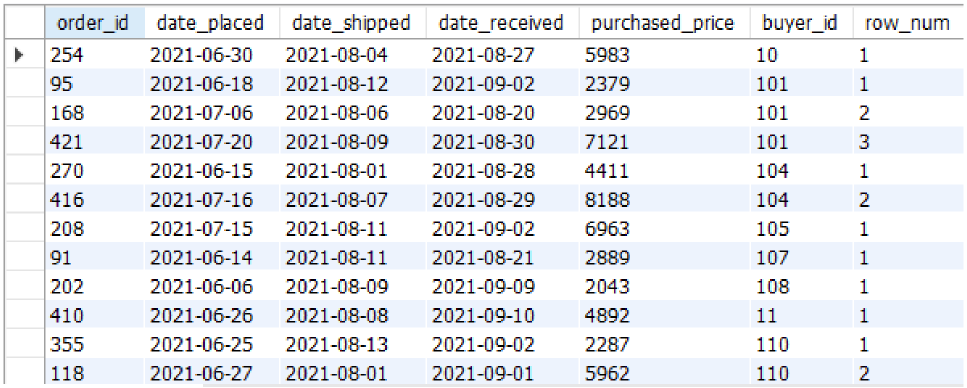
```
WITH cte_corr AS
(SELECT  (count(*) * sum(x * y) - sum(x) * sum(y)) /
      (sqrt(count(*) * sum(x * x) - sum(x) * sum(x)) * sqrt(count(*) * sum(y * y) - sum(y) *
sum(y)))
      AS correlation_perimeter_price,
      (count(*) * sum(z * y) - sum(z) * sum(y)) /
      (sqrt(count(*) * sum(z * z) - sum(z) * sum(z)) * sqrt(count(*) * sum(y * y) - sum(y) * sum(y)))
      AS correlation_area_price
   FROM (SELECT price as y, length_cm + height_cm + width_cm as x, length_cm * height_cm *
width_cm z FROM artwork) derived)
SELECT * FROM cte_corr;
```

Query # 2: What is the average time a buyer waits before making their next purchase? Compare dates of placing orders and find the average time between the orders. Show the results for buyers that have placed at least 2 orders.

We want to figure out the average time between a buyer's first purchase and their next.

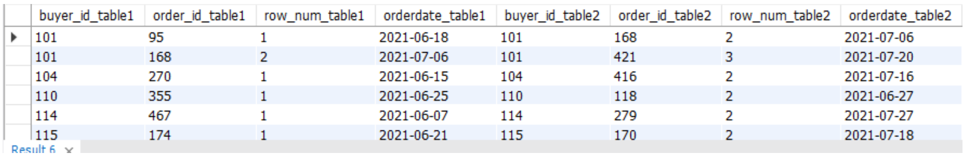
In order to do that, our code follows the steps below:

a. Orders records for each buyer by the date that they placed their order in a temporary table.
b. For a buyer we increment row numbers for the next order, otherwise we reset the row number to 1.



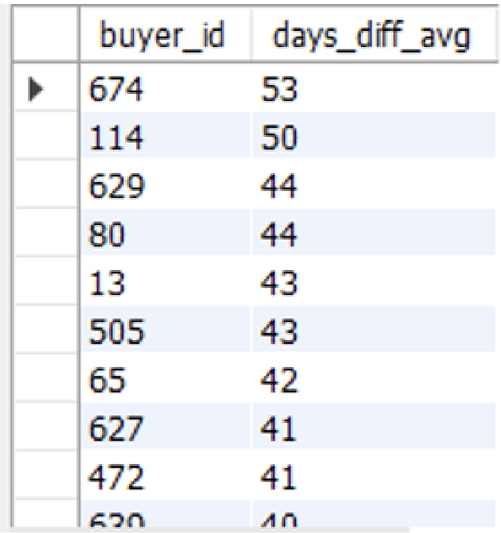
order_id	date_placed	date_shipped	date_received	purchased_price	buyer_id	row_num
254	2021-06-30	2021-08-04	2021-08-27	5983	10	1
95	2021-06-18	2021-08-12	2021-09-02	2379	101	1
168	2021-07-06	2021-08-06	2021-08-20	2969	101	2
421	2021-07-20	2021-08-09	2021-08-30	7121	101	3
270	2021-06-15	2021-08-01	2021-08-28	4411	104	1
416	2021-07-16	2021-08-07	2021-08-29	8188	104	2
208	2021-07-15	2021-08-11	2021-09-02	6963	105	1
91	2021-06-14	2021-08-11	2021-08-21	2889	107	1
202	2021-06-06	2021-08-09	2021-09-09	2043	108	1
410	2021-06-26	2021-08-08	2021-09-10	4892	11	1
355	2021-06-25	2021-08-13	2021-09-02	2287	110	1
118	2021-06-27	2021-08-01	2021-09-01	5962	110	2

c. We have two copies as temporary tables (because temporary tables can't be accessed twice in a query in MySQL). We then join the two tables such that for a buyer, we match the 1st order with the second between tables. An inner join eliminates any buyers that did not place a second order.



buyer_id_table1	order_id_table1	row_num_table1	orderdate_table1	buyer_id_table2	order_id_table2	row_num_table2	orderdate_table2
101	95	1	2021-06-18	101	168	2	2021-07-06
101	168	2	2021-07-06	101	421	3	2021-07-20
104	270	1	2021-06-15	104	416	2	2021-07-16
110	355	1	2021-06-25	110	118	2	2021-06-27
114	467	1	2021-06-07	114	279	2	2021-07-27
115	174	1	2021-06-21	115	170	2	2021-07-18

d. We simply group by on buyer and find average difference (in days) between the first and second order’s dates. Then we sort on days descending.



buyer_id	days_diff_avg
674	53
114	50
629	44
80	44
13	43
505	43
65	42
627	41
472	41
620	40

Code:

```
DROP TEMPORARY TABLE IF EXISTS dt;
CREATE TEMPORARY TABLE dt
SELECT * FROM orders order by buyer_id, date_placed;
DROP TEMPORARY TABLE IF EXISTS cte_row_temp_1;
CREATE TEMPORARY TABLE cte_row_temp_1
SELECT *, ROW_NUMBER() OVER(PARTITION BY buyer_id) AS row_num FROM dt ORDER BY
buyer_id ASC, date_placed ASC;
DROP TEMPORARY TABLE IF EXISTS cte_row_temp_2;
CREATE TEMPORARY TABLE cte_row_temp_2
SELECT * FROM cte_row_temp_1;

SELECT r1.buyer_id, DATEDIFF(r2.date_placed, r1.date_placed) days_diff_avg

FROM cte_row_temp_1 AS r1 join cte_row_temp_2 AS r2 ON r1.row_num+1 = r2.row_num
AND r1.buyer_id = r2.buyer_id GROUP BY 1 ORDER BY 2 DESC;
```

Insights gained:

- Setting out the business scenario and goals at the beginning of the project helps greatly to define the scope of the database and identify entities that need to be included in the ERD. Correctly defining the entities at the ERD phase, eliminates the need for normalization.
- We have become more proficient in data manipulation & managing data changes. We used transactions to ensure our data changes were correct and only then we COMMIT these changes permanently.
- We have learned about Common Table Expressions, Stored Procedures, Storing Variables, Setting Constraints, Temporary Tables & managing grants/rights.
- This project highlighted why it is very important to ensure everyone is on the same page, and communicate efficiently at every stage of this project, since every section is interconnected with other sections.
- The MySQL Export Dump file exports views & procedures with statement ‘definer = <>’ which required views & stored procedures to be altered after importing. This was manually changed in the .sql dump file itself so that any user can import the database by simply running all the sql script.