INTRODUCTION

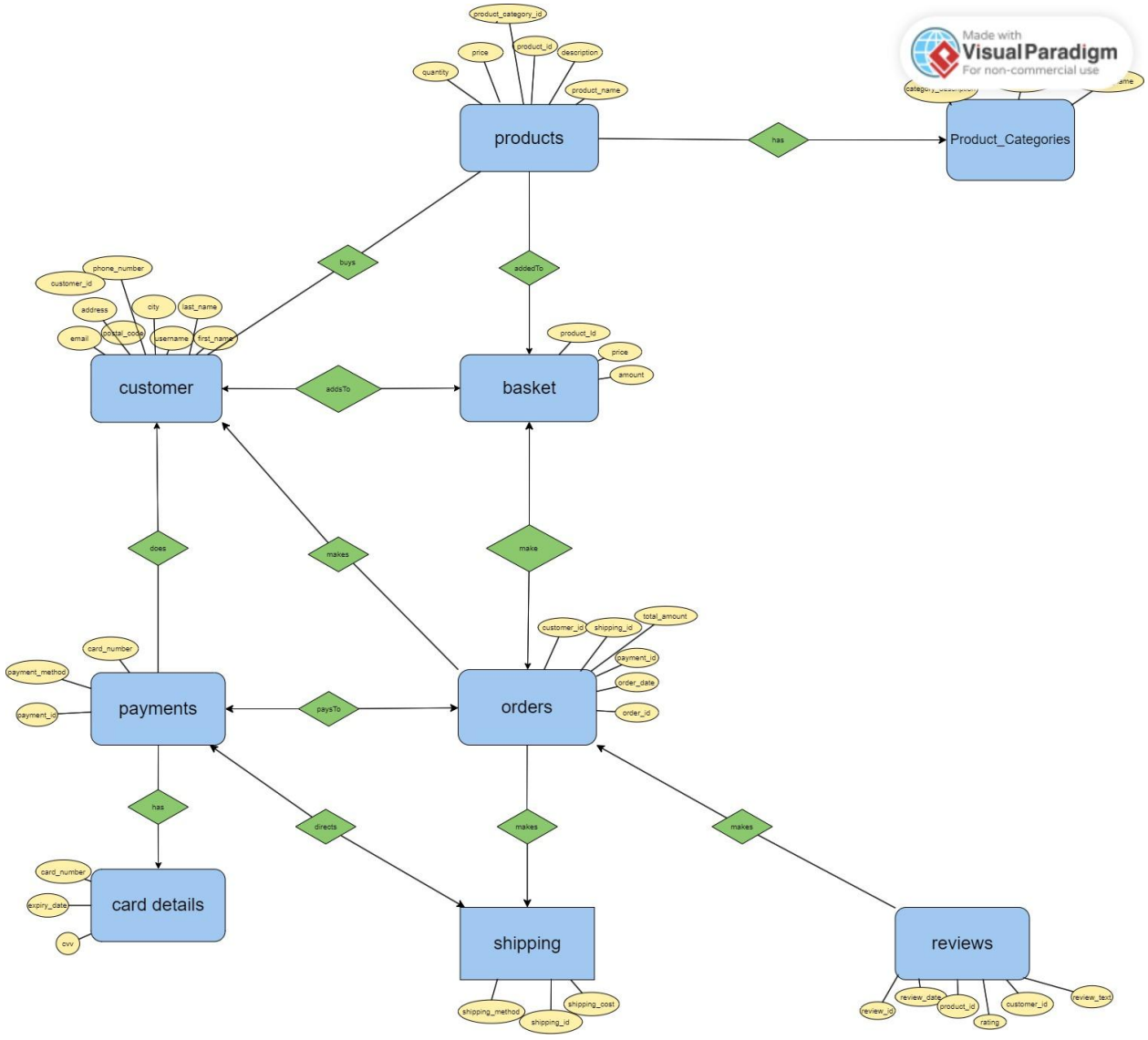
E-commerce system - online shop for kid’s clothes (ex. bambino_shop.almaty)

Welcome to this project report E-Commerce Systems - Online Store for Kid's Clothing. Why we chose this particular e-commerce system, firstly, because In today's digital age, e-commerce has become an integral part of retail, and online shopping has become a commonplace for many consumers. The goal of this project was to develop an e-commerce system specifically designed to meet the needs of parents who want to buy children's clothing.

The online children's clothing store offers a wide range of products for different age groups, genders and occasions. The system has been designed to be user friendly and easy to navigate, making it easy for customers to view products, add items to their shopping cart and make secure payments.

This report provides an overview of the project, including its objectives, scope, and methodology. It also contains the details of ER diagram, normalization and queries.

ERD



*Customer table has many-to-many relationships because multiple Customers can buy multiple Products. And multiple Products can be bought by multiple Customers

*Customers table has a one-to-many relationship with the Orders table, as each customer can have multiple orders but each order belongs to a single customer.

*The Orders table has a one-to-one relationship with the Basket table. This would mean that each order in the Orders table is associated with only one basket in the Basket table, and each basket is associated with only one order.

*Products table has a many-to-one relationship with the Product_Categories table, as each product belongs to a single category but each category can contain multiple products.

*The relationship between a Basket and Products is one-to-many, as a single basket can contain multiple products.

*Orders table has a one-to-one relationship with the Payments table, as each order can have a single payment and each payment belongs to a single order.

*Orders table has a many-to-one relationship with the Shipping table, as each order can have a single shipping method but each shipping method can be used for multiple orders.

*The relationship between Customers and Baskets is one-to-one. This is because a customer can have one baskets, and a basket can be associated with one customers

* The Orders table and Reviews table have a one-to-many relationship because one order can have multiple reviews, but each review is associated with only one order.

* The relationship between Payments and Shipping tables is one-to-one. This is because one payments can be associated with one shipment

* The relationship between Payments and Customers tables are many-to-one, where multiple payments can be associated with a single customer.

* Payments table has many-to-one relationship with the Card_details table, because each payment in the Payments table can be associated with one card details in the Card_details table, but each card detail can be associated with multiple payment

PROCESS CREATING TABLE AND INSERTING DATA

TABLES

1)Customers:

```
CREATE TABLE Customers (
  customer_id   INTEGER PRIMARY KEY,
  first_name    VARCHAR2(50) NOT NULL,
  last_name     VARCHAR2(50) NOT NULL,
  email         VARCHAR2(100) NOT NULL UNIQUE,
  phone_number  VARCHAR2(20),
  address       VARCHAR2(200),
  city          VARCHAR2(50)
  postal_code   VARCHAR2(20)
);
```

2)Orders:

```
CREATE TABLE Orders (
  order_id      INTEGER PRIMARY KEY,
  customer_id   INTEGER NOT NULL,
  order_date    DATE NOT NULL,
  total_amount  NUMBER(10,2) NOT NULL,
```

```
  payment_id    INTEGER NOT NULL,
  shipping_id   INTEGER NOT NULL,
  CONSTRAINT fk_orders_customers
    FOREIGN KEY (customer_id)
    REFERENCES Customers(customer_id),
  CONSTRAINT fk_orders_payments
    FOREIGN KEY (payment_id)
    REFERENCES Payments(payment_id),
  CONSTRAINT fk_orders_shipping
    FOREIGN KEY (shipping_id)
    REFERENCES Shipping(shipping_id)
);
```

3) Products:

```
CREATE TABLE Products (
  product_id      INTEGER PRIMARY KEY,
  product_name    VARCHAR2(100) NOT NULL UNIQUE,
  description     VARCHAR2(2000),
  price           NUMBER(10,2) NOT NULL,
  quantity        INTEGER NOT NULL,
  product_category_id INTEGER NOT NULL,
```

```
  CONSTRAINT fk_products_product_category
    FOREIGN KEY (product_category_id)
    REFERENCES Product_Categories(product_category_id)
);
```

4) Product categories:

```
CREATE TABLE Product_Categories (
  product_category_id   INTEGER PRIMARY KEY,
  category_name         VARCHAR2(100) NOT NULL UNIQUE,
  category_description  VARCHAR2(2000)
);
```

5) Payments:

```
CREATE TABLE Payments (
  payment_id     INTEGER PRIMARY KEY,
  payment_method VARCHAR2(50) NOT NULL,
  card_number    VARCHAR2(50) NOT NULL UNIQUE,
 );
```

6) Shipping:

```
CREATE TABLE Shipping (
```

```
  shipping_id     INTEGER PRIMARY KEY,
  shipping_method VARCHAR2(50) NOT NULL UNIQUE,
  shipping_cost   NUMBER(10,2) NOT NULL
);
```

7) Reviews:

```
CREATE TABLE Reviews (
  review_id       INTEGER PRIMARY KEY,
  product_id      INTEGER NOT NULL,
  customer_id     INTEGER NOT NULL,
  review_text     VARCHAR2(2000),
  review_date     DATE NOT NULL,
  rating          NUMBER(1,0) NOT NULL,
  CONSTRAINT fk_reviews_products
    FOREIGN KEY (product_id)
    REFERENCES Products(product_id),
  CONSTRAINT fk_reviews_customers
    FOREIGN KEY (customer_id)
    REFERENCES Customers(customer_id)
);
```

8)Card_details:

```
CREATE TABLE Card_details(
  card_number    VARCHAR2(50) NOT NULL UNIQUE,
  expiry_date    VARCHAR2(10) NOT NULL,
  cvv            VARCHAR2(10) NOT NULL
);
```

9)**Backet**:

```
CREATE TABLE Backet(
Product_id INTEGER NOT NULL,
Price NUMBER(10,2) NOT NULL,
Amount INTEGER NOT NULL);
```

Normalization

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update, and Deletion Anomalies. Normalization rules divide larger tables into smaller tables and link them using relationships.

Normal Forms:

There are several normal forms in database normalization, including first normal form (1NF), second normal form (2NF), and third normal form (3NF). Each normal form builds on the previous one and has specific criteria that must be met to achieve it.

- First Normal Form (1NF):

The first normal form requires that a table must have only atomic values, meaning each column should contain only one piece of information and that there should be no repeating groups or arrays. Additionally, there should be no duplicate data in the table.

- Second Normal Form (2NF):

The second normal form requires that a table should meet the criteria for 1NF and should have a single candidate key. Furthermore, all non-key attributes should be functionally dependent on the candidate key. In other words, all attributes in the table must relate to the primary key of the table.

- Third Normal Form (3NF):

The third normal form requires that a table should meet the criteria for 2NF and should have no transitive dependencies between non-key attributes. All attributes must depend solely on the primary key, which means that there are no non-key attributes that depend on other non-key attributes.

Let’s discover tables of online-shop database:

1)Customers

Functional dependencies for the Customers table:

customer_id -> first_name, last_name, email, phone_number, address, city, postal_code

1NF (First Normal Form):

- The table has a primary key column (customer_id) which has unique values for each row.
- Each column in the table contains atomic values.
- There are no repeating groups of data in any of the columns.

2NF (Second Normal Form):

- All non-key attributes (**first_name, last_name, email, phone_number, address, city, postal_code**) functionally dependent on the candidate key.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes. All attributes depend solely on the primary key (customer_id), which means that there are no non-key attributes that depend on other non-key attributes.

2)Orders

Functional dependencies for the **Orders** table:

order_id -> customer_id, order_date, total_amount, payment_id, shipping_id

1NF (First Normal Form):

- The table has a primary key column (**order_id**) which has unique values for each row.

- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes (**customer_id, order_date, total_amount, payment_id, shipping_id**) functionally dependent on the candidate key.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.

//comments {actually total amount can be depended on payment id, but from our business logic in this table total amount is only total cost or orders, the cost which we will get with payment id can differ from total amount (bonus etc). Same to order date (can diff from shipping date) and total amount (dest, weight) with shipping id.

3)Products

Functional dependencies for the **Products** table:

product_id -> product_name, description, price, quantity, product_category_id

1NF (First Normal Form):

- The table has a primary key column (**product_id**) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes (**product_name, description, price, quantity, product_category_id**) functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

//comment: product_name cannot identify the description, price and quantity since there can be different toys with different sizes but with the same name. So it will depend from only primary key.

4)Product categories:

product_category_id -> category_name, category_description

1NF (First Normal Form):

- The table has a primary key column (**product_category_id**) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes (**category_name, category_description**) functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

5)Payments

payment_id -> payment_method , card_number

1NF (First Normal Form):

- The table has a primary key column (payment_id) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes (payment_method , card_number) functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

6)Shipping

shipping_id -> shipping_method , shipping_cost

1NF (First Normal Form):

- The table has a primary key column (shipping_id) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes (shipping_method , shipping_cost functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

7) Reviews

review_id -> review_text, review_date, rating, review_id,product_id, customer_id

1NF (First Normal Form):

- The table has a primary key column (**review_id**) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

8)Card_details

card_number -> expiry_date **,** cvv

1NF (First Normal Form):

- The table has a primary key column (**card_number**) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

9)Backet

Product_id ->Price ,Amount

1NF (First Normal Form):

- The table has a primary key column (**Product_id**) which has unique values for each row.
- Each column in the table contains atomic values.

2NF (Second Normal Form):

- All non-key attributes functionally dependent on the candidate key. No partial dependencies.

3NF (Third Normal Form):

- The table is also in 3NF since there are no transitive dependencies between non-key attributes.Since all non primary keys depended only from primary key.

QUERIES

INSERTING DATA

1) Customers

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (101, 'Steven', 'Taylor', 'steven.taylor@example.com', '855-585-5555', '246 Birch St', 'Almaty', '050037');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (102, 'Samantha', 'White', 'samantha.white@example.com', '545-785-5055', '579 Elm St', 'Astana', '246801');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (103, 'Mary', 'Davis', 'mary.davis@example.com', '455-595-5055', '321 Pine St', 'Aktau', '060024');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (104, 'John', 'Doe', 'john.doe@example.com', '505-515-5055', '123 Main St', 'Aktobe', '123450');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (105, 'Jane', 'Smith', 'jane.smith@example.com', '255-515-5505', '456 Elm St', 'Almaty', 050037);

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (106, 'Bob', 'Johnson', 'bob.johnson@example.com', '355-955-5585', '789 Oak St', 'Talgar', '543218');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (107, 'Karen', 'Wilson', 'karen.wilson@example.com', '755-105-5895', '987 Cedar St', 'Almaty', '050037');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (108, 'Amy', 'Brown', 'amy.brown@example.com', '955-555-5555', '135 Oak Ave', 'Kaskelen', '864204');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (109, 'Tom', 'Garcia', 'tom.garcia@example.com', '555-535-7855', '468 Pine St', 'Astana', '246801');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address, city, postal_code) VALUES (110, 'David', 'Lee', 'david.lee@example.com', '655-575-5325', '654 Maple St', 'Shymkent', '234560');

2) Orders

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (1, 101, '01/04/2022', 50.00, 1, 1);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES(2, 102, '05/05/2022', 25.00, 2, 2);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (3, 103, '09/08/2022', 75.00, 1, 1);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES(4, 104, '10/10/2022', 100.00, 3, 3);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (5, 105, '06/12/2022', 60.00, 2, 2);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (6, 106, '12/22/2022', 45.00, 1, 1);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (7, 107, '02/02/2023', 80.00, 3, 3);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES(8, 108, '03/05/2023', 35.00, 2, 2);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (9, 109, '04/05/2023', 95.00, 1, 1);

INSERT INTO Orders (order_id, customer_id, order_date, total_amount, payment_id, shipping_id) VALUES (10, 110, '05/25/2023', 50.00, 3, 3);

3) Products

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (1, 'Kids T-Shirt', 'A comfortable and stylish t-shirt for kids', 12.99, 50, 1);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (2, 'Kids Sneakers', 'High-quality sneakers for kids', 39.99, 30, 2);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (3, 'Building Blocks Set', 'A fun and educational building blocks set for kids', 29.99, 20, 3);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (4, 'Coloring Book', 'An engaging coloring book for kids', 9.99, 100, 4);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (5, 'Art Supplies Kit', 'A comprehensive art supplies kit for kids', 24.99, 10, 5);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (6, 'Backpack', 'A durable and spacious backpack for school', 34.99, 40, 6);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (7, 'Bedding Set', 'A cozy and cute bedding set for kids', 79.99, 15, 7);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (8, 'Kid-Sized Table and Chairs', 'A set of table and chairs perfect for playrooms and bedrooms', 99.99, 5, 8);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (9, 'Hair Accessories Set', 'A set of fun and colorful hair accessories for kids', 14.99, 25, 9);

INSERT INTO Products (product_id, product_name, description, price, quantity, product_category_id) VALUES (10, 'Outdoor Play Set', 'A set of equipment and supplies for outdoor activities for kids', 69.99, 8, 10);

4)Product_Categories

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (1, 'Clothing', 'A variety of clothes for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (2, 'Shoes', 'Comfortable and stylish shoes for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (3, 'Toys', 'Fun and educational toys for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (4, 'Books', 'Engaging and educational books for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (5, 'Arts and Crafts', 'Art supplies and craft kits for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (6, 'School Supplies', 'Essential supplies for kids going to school');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (7, 'Bedding', 'Comfortable bedding for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (8, 'Furniture', 'Kid-sized furniture for playrooms and bedrooms');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (9, 'Accessories', 'Fun and cute accessories for kids');

INSERT INTO Product_Categories (product_category_id, category_name, category_description) VALUES (10, 'Outdoor Gear', 'Equipment and supplies for outdoor activities for kids');

5) Payments

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(1, 'Visa', '7890123456789012');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES (2, 'MasterCard', '8901234567890123');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(3, 'American Express', '9012345678901234');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(4, 'Discover', '1234567890123456');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(5, 'Visa', '2345678901234567');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(6, 'MasterCard', '6789012345678901');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(7, 'American Express', '3456789012345678');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(8, 'Discover', '4567890123456789');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES(9, 'Visa', '5678901234567890');

INSERT INTO Payments(payment_id, payment_method, card_number) VALUES (10, 'MasterCard', '6789012345678901');

6) Shipping

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (1, 'Standard Shipping', 6.99);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (2, 'Express Shipping', 12.99);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (3, 'Free Shipping', 0.00);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (4, 'Standard Shipping', 5.99);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (5, 'Standard Shipping', 5.99);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (6, 'Local Delivery', 3.55);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (7, '2-Day Shipping', 8.99);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (8, 'Next Day Shipping', 11.99);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (9, 'In-Store Pickup', 0.00);

INSERT INTO Shipping (shipping_id, shipping_method, shipping_cost) VALUES (10, 'White Glove Delivery', 7.99);

7) Reviews

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (1, 1, 101, 'My kid loves this shirt!', TO_DATE('2023-03-30', 'YYYY-MM-DD'), 5);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (2, 2, 102, 'Great sneakers for active kids', TO_DATE('2023-03-28', 'YYYY-MM-DD'), 4);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (3, 3, 103, 'My kids have been playing with these blocks for hours', TO_DATE('2023-03-25', 'YYYY-MM-DD'), 5);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (4, 4, 104, 'Good quality coloring book for young artists', TO_DATE('2023-03-22', 'YYYY-MM-DD'), 4);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (5, 5, 105, 'This art supplies kit has everything my kids need to get creative', TO_DATE('2023-03-20', 'YYYY-MM-DD'), 5);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (6, 6, 106, 'Sturdy backpack that fits all of my kid''s school supplies', TO_DATE('2023-03-18', 'YYYY-MM-DD'), 4);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (7, 7, 107, 'This bedding set is adorable and comfortable', TO_DATE('2023-03-16', 'YYYY-MM-DD'), 5);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (8, 8, 108, 'Perfect for my kid''s playroom!', TO_DATE('2023-03-14', 'YYYY-MM-DD'), 4);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (9, 9, 109, 'My daughter loves these hair accessories', TO_DATE('2023-03-12', 'YYYY-MM-DD'), 5);

INSERT INTO Reviews (review_id, product_id, customer_id, review_text, review_date, rating) VALUES (10, 10, 110, 'Great outdoor play set for my kids', TO_DATE('2023-03-10', 'YYYY-MM-DD'), 4);

7)Card_details

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES

('1234567890123456', '12/25', '123'),

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('2345678901234567', '01/24', '234');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('3456789012345678', '03/26', '345');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('4567890123456789', '06/27', '456');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('5678901234567890', '09/28', '567');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('6789012345678901', '11/23', '678');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('7890123456789012', '02/25', '789';

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('8901234567890123', '07/24', '890');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('9012345678901234', '09/26', '901');

INSERT INTO Card_details (card_number, expiry_date, cvv) VALUES ('0123456789012345', '05/27', '012');

8)Backet

INSERT INTO Backet(product_id, price, amount) VALUES (1, 12.99, 2);

INSERT INTO Backet(product_id, price, amount) VALUES (2, 39.99, 1);

INSERT INTO Backet(product_id, price, amount) VALUES (3, 29.99, 4);

INSERT INTO Backet(product_id, price, amount) VALUES (4, 9.99, 3);

INSERT INTO Backet(product_id, price, amount) VALUES (5, 2499, 2);

INSERT INTO Backet(product_id, price, amount) VALUES (6, 34.99, 3);

INSERT INTO Backet(product_id, price, amount) VALUES (7, 79.99, 5);

INSERT INTO Backet(product_id, price, amount) VALUES (8, 99.99, 1);

INSERT INTO Backet(product_id, price, amount) VALUES (9, 14.99, 6);

INSERT INTO Backet(product_id, price, amount) VALUES (10, 69.99, 7);

1) Procedure which does group by information

```
CREATE OR REPLACE PROCEDURE group_by_category
IS
BEGIN
   FOR rec IN (
      SELECT PRODUCT_CATEGORY_ID, COUNT(*) AS num_products
```

```
    FROM products
    GROUP BY PRODUCT_CATEGORY_ID
  )
  LOOP
DBMS_OUTPUT.PUT_LINE('Category: ' || rec.PRODUCT_CATEGORY_ID || ', Number of Products: ' || rec.num_products);
  END LOOP;
END;
```

How to get the result:

```
BEGIN
  group_by_category;
END;
```

2) Function which counts the number of records

```
CREATE OR REPLACE FUNCTION totalReviews(
  date_from DATE DEFAULT SYSDATE-7,
  date_to DATE DEFAULT SYSDATE)
RETURN NUMBER
IS
  total NUMBER(10) := 0;
```

```
BEGIN
   SELECT COUNT(*)
   INTO total
   FROM Reviews
   WHERE review_date BETWEEN date_from AND date_to;

   RETURN total;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No reviews found for the selected period.');
      RETURN NULL;
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error’);
      RETURN NULL;
END;
```

How to call a function?

```
DECLARE
   review_count NUMBER;
BEGIN
   review_count := totalReviews(
```

```
        date_from => TO_DATE('03/28/2023', 'MM/DD/YYYY'),
        date_to   => TO_DATE('03/30/2023', 'MM/DD/YYYY')
    );

    DBMS_OUTPUT.PUT_LINE('Number of reviews between 03/28/2023 and 03/30/2023: ' || review_count);
END;
```

3) Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

```
CREATE OR REPLACE PROCEDURE update_product_price(
    in_product_id INTEGER,
    in_price NUMBER
)
IS
BEGIN
    UPDATE Products
    SET price = in_price
    WHERE product_id = in_product_id;

    DBMS_OUTPUT.PUT_LINE('Number of rows cool updated: ' || SQL%ROWCOUNT);
```

```
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No data found for the specified product ID.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error’);
        ROLLBACK;
END;
```

Example:

```
BEGIN
    update_product_price(in_product_id => 5, in_price => 9.99);
END;
```

4) Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters

```
CREATE OR REPLACE TRIGGER trigger_products_description
BEFORE INSERT OR UPDATE ON Products
FOR EACH ROW
DECLARE
    description_short EXCEPTION;
```

```
BEGIN
   IF LENGTH(:NEW.description) < 5 THEN
      RAISE description_short;
   END IF;
EXCEPTION
   WHEN description_short THEN
         dbms_output.put_line('Product description must be at least 5 characters long.');
END;
/
```

To check:

```
insert into products
values(19,'Doll','l',15.99,5,5)
```

—----------

```
CREATE OR REPLACE TRIGGER trigger_customers_email
BEFORE INSERT OR UPDATE ON Customers
FOR EACH ROW
```

```
DECLARE
   email_short EXCEPTION;
BEGIN
   IF LENGTH(:NEW.email) < 5 THEN
      RAISE email_short;
   END IF;
EXCEPTION
   WHEN email_short THEN
      dbms_output.put_line( 'Email must be at least 5 characters long.');
END;
```

5) Create a trigger before insert on any entity which will show the current number of rows in the table

```
CREATE OR REPLACE TRIGGER products_before_insert
BEFORE INSERT ON products
FOR EACH ROW
DECLARE
   row_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO row_count FROM products;
```

```
    DBMS_OUTPUT.PUT_LINE('Current number of rows in products table: ' ||
row_count);


EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('No data found in products table.');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: ');
END;
```