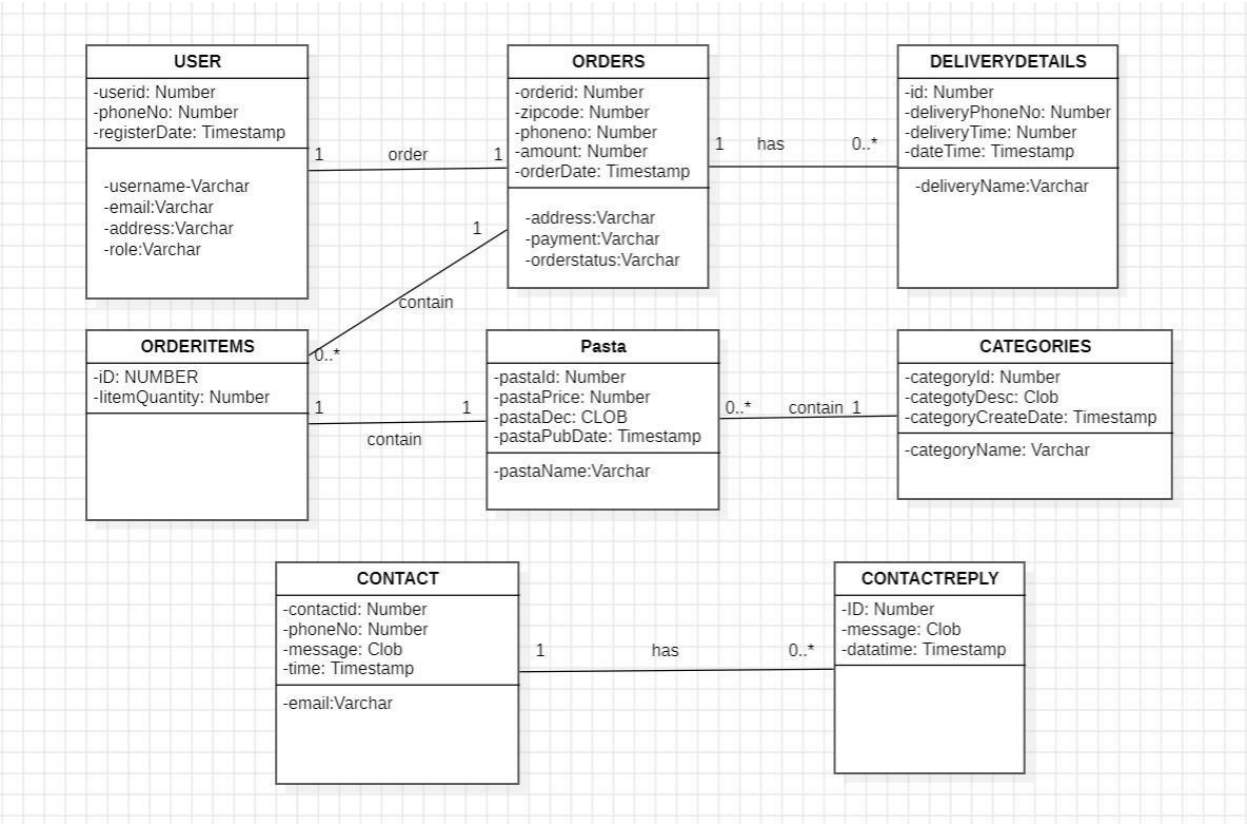
CET341 Assignment 2

Student Name – Nyi Zaw

Student Id – (bi55is)

Student no-239349271
Computer System Engineering

Task One:


USER
-userid: Number
-phoneNo: Number
-registerDate: Timestamp
-username-Varchar
-email:Varchar
-address:Varchar
-role:Varchar
1
order
1
ORDERS
-orderid: Number
-zipcode: Number
-phoneno: Number
-amount: Number
-orderDate: Timestamp
-address:Varchar
-payment:Varchar
-orderstatus:Varchar
1
has
0..*
DELIVERYDETAILS
-id: Number
-deliveryPhoneNo: Number
-deliveryTime: Number
-dateTime: Timestamp
-deliveryName:Varchar
1
contain
0..*
ORDERITEMS
-iD: NUMBER
-litemQuantity: Number
1
contain
1
Pasta
-pastaId: Number
-pastaPrice: Number
-pastaDec: CLOB
-pastaPubDate: Timestamp
-pastaName:Varchar
0..*
contain
1
CATEGORIES
-categoryId: Number
-categotyDesc: Clob
-categoryCreateDate: Timestamp
-categoryName: Varchar
CONTACT
-contactid: Number
-phoneNo: Number
-message: Clob
-time: Timestamp
-email:Varchar
1
has
0..*
CONTACTREPLY
-ID: Number
-message: Clob
-datatime: Timestamp

Task Two:

Categories table is created.

```
CREATE TABLE categories (
    categorieid          NUMBER NOT NULL,
    categoriename        VARCHAR(255) NOT NULL,
    categoriedesc        CLOB NOT NULL,
    categoriecreatedate  TIMESTAMP DEFAULT systimestamp NULL
);
```

```
*Action:

Table CATEGORIES created.
```

Insert data to categories table.

```
INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(1, 'Spaghetti Carbonara', 'Indulge in our creamy and savory Spaghetti Carbonara, made with pancetta, egg, and Parm

INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(2, 'Penne Arrabbiata', 'Savor the spicy and robust flavors of our Penne Arrabbiata, featuring a tangy tomato sauce

INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(3, 'Fettuccine Alfredo', 'Enjoy the rich and creamy taste of our Fettuccine Alfredo, made with a smooth Alfredo sa

INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(4, 'Vegetarian Pasta', 'Delight in our Vegetarian Pasta, a meat-free option filled with colorful vegetables, tofu,

INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(5, 'Side Dishes', 'Complement your pasta with our selection of side dishes, including bruschetta, garlic bread, an

INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(6, 'Pasta Bowls', 'Explore our variety of pasta bowls, each crafted with care to deliver authentic Italian flavors
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.257 seconds

```
1 row inserted.

1 row inserted.
```

Create Contact table

```
INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(7, 'Specialty Pasta', 'Indulge in our specialty pasta creations, featuring unique flavor combinations and premiu

INSERT INTO categories (categorieId, categorieName, categorieDesc, categorieCreateDate) VALUES
(8, 'Drinks', 'Quench your thirst with our selection of refreshing drinks, including traditional Italian sodas, w

DROP TABLE contact;

CREATE TABLE contact (
    contactid  NUMBER NOT NULL,
    userid     NUMBER NOT NULL,
    email      VARCHAR(35) NOT NULL,
    phoneno    NUMBER NOT NULL,
    orderid    NUMBER NOT NULL,
    message    CLOB NOT NULL,
    time       TIMESTAMP DEFAULT systimestamp NULL
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.053 seconds

```
Table CONTACT created.
```

Insert data to Contact table

```
CREATE TABLE contact (
    contactid  NUMBER NOT NULL,
    userid     NUMBER NOT NULL,
    email      VARCHAR(35) NOT NULL,
    phoneno    NUMBER NOT NULL,
    orderid    NUMBER NOT NULL,
    message    CLOB NOT NULL,
    time       TIMESTAMP DEFAULT systimestamp NULL
);

INSERT INTO contact (contactId, userId, email, phoneNo, orderId, message, time) VALUES
(1, 2, 'nicky@gmail.com', 1234567890, 1, 'Nice You guys are doing great..', TIMESTAMP '2023-01-30 20:39:4
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.051 seconds

```
1 row inserted.
```

Create Contact reply Table

```
CREATE TABLE contactreply (
    id          NUMBER NOT NULL,
    contactid   NUMBER NOT NULL,
    userid      NUMBER NOT NULL,
    message     CLOB NOT NULL,
    datetime    TIMESTAMP DEFAULT systimestamp NULL
);

INSERT INTO contactreply (id, contactId, userId, message, datetime) VALUES
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

Task completed in 0.05 seconds

```
Table CONTACTREPLY created.
```

Insert data to Contact reply table

```
CREATE TABLE contactreply (
    id          NUMBER NOT NULL,
    contactid   NUMBER NOT NULL,
    userid      NUMBER NOT NULL,
    message     CLOB NOT NULL,
    datetime    TIMESTAMP DEFAULT systimestamp NULL
);

INSERT INTO contactreply (id, contactId, userId, message, datetime) VALUES
(1, 1, 2, 'Thank you for shopping with us', TIMESTAMP '2023-01-30 20:40:08');
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Quer

Task completed in 0.099 seconds

```
Table CONTACTREPLY created.


1 row inserted.
```

Create Delivery Details table

```
CREATE TABLE deliverydetails (
    id                  NUMBER NOT NULL,
    orderid             NUMBER NOT NULL,
    deliveryname     VARCHAR(35) NOT NULL,
    deliveryphoneno  NUMBER NOT NULL,
    deliverytime        NUMBER NOT NULL,
    datetime            TIMESTAMP DEFAULT systimestamp NULL
);
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3

Task completed in 0.056 seconds

```
Table DELIVERYDETAILS created.
```

Insert data to Delivery Details table

```
INSERT INTO deliverydetails (id, orderId, deliveryName, deliveryPhoneNo, deliveryTime, dateTime) VALUES
(1, 1, 'Alice Johnson', 9988776655, 40, TIMESTAMP '2023-06-10 10:15:00');

INSERT INTO deliverydetails (id, orderId, deliveryName, deliveryPhoneNo, deliveryTime, dateTime) VALUES
(2, 2, 'Bob Smith', 8877665544, 45, TIMESTAMP '2023-06-11 12:30:00');

INSERT INTO deliverydetails (id, orderId, deliveryName, deliveryPhoneNo, deliveryTime, dateTime) VALUES
(3, 3, 'Charlie Brown', 7766554433, 50, TIMESTAMP '2023-06-12 14:45:00');

INSERT INTO deliverydetails (id, orderId, deliveryName, deliveryPhoneNo, deliveryTime, dateTime) VALUES
(4, 4, 'Daisy Miller', 6655443322, 35, TIMESTAMP '2023-06-13 16:00:00');

INSERT INTO deliverydetails (id, orderId, deliveryName, deliveryPhoneNo, deliveryTime, dateTime) VALUES
(5, 8, 'Eddie Wilson', 5544332211, 60, TIMESTAMP '2023-06-14 18:15:00');
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.286 seconds

```
1 row inserted.

1 row inserted.
```

Create order Items table

```
CREATE TABLE orderitems (
  id NUMBER NOT NULL,
  orderId NUMBER NOT NULL,
  pastaId NUMBER NOT NULL,
  itemQuantity NUMBER NOT NULL
);

INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VAI
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Que

Task completed in 0.033 seconds

```
Table ORDERITEMS created.
```

Insert data to order items table

```
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (1, 1, 1, 2);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (2, 2, 1, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (3, 2, 2, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (4, 2, 3, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (5, 3, 1, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (6, 3, 2, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (7, 3, 3, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (8, 3, 4, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (9, 3, 5, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (10, 3, 6, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (11, 3, 7, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (12, 4, 13, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (13, 4, 14, 1);
INSERT INTO orderitems (id, orderId, pastaId, itemQuantity) VALUES (14, 4, 15, 1);
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Resul

Task completed in 1.208 seconds

```
1 row inserted.

1 row inserted.
```

Create order table

```
DROP TABLE orders;

CREATE TABLE orders (
  orderId NUMBER NOT NULL,
  userId NUMBER NOT NULL,
  address VARCHAR2(255) NOT NULL,
  zipCode NUMBER NOT NULL,
  phoneNo NUMBER NOT NULL,
  amount NUMBER NOT NULL,
  paymentMode VARCHAR2(1) NOT NULL,
  orderStatus VARCHAR2(1) NOT NULL,
  orderDate TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.042 seconds

```
Table ORDERS created.
```

Insert data to order table

```
INSERT INTO orders (orderId, userId, address, zipCode, phoneNo, amount, paymentMode,
VALUES (1, 2, 'Rome, Italy', 00100, 39061234567, 198, '1', '1', TIMESTAMP '2023-05-0

INSERT INTO orders (orderId, userId, address, zipCode, phoneNo, amount, paymentMode,
VALUES (2, 2, 'Milan, Italy', 20100, 390212345678, 447, '1', '2', TIMESTAMP '2023-05-

INSERT INTO orders (orderId, userId, address, zipCode, phoneNo, amount, paymentMode,
VALUES (3, 2, 'Naples, Italy', 80100, 390812345678, 1463, '1', '3', TIMESTAMP '2023-

INSERT INTO orders (orderId, userId, address, zipCode, phoneNo, amount, paymentMode,
VALUES (4, 2, 'Turin, Italy', 10100, 390112345678, 697, '1', '4', TIMESTAMP '2023-05-

INSERT INTO orders (orderId, userId, address, zipCode, phoneNo, amount, paymentMode,
VALUES (5, 2, 'Palermo, Italy', 90100, 390912345678, 99, '1', '1', TIMESTAMP '2023-0

INSERT INTO orders (orderId, userId, address, zipCode, phoneNo, amount, paymentMode,
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Resu

Task completed in 0.359 seconds

```
1 row inserted.

1 row inserted.
```

Create pasta table

```
CREATE TABLE pasta (
  pastaId NUMBER NOT NULL,
  pastaName varchar(255) NOT NULL,
  pastaPrice NUMBER NOT NULL,
  pastaDesc CLOB NOT NULL,
  pastaCategorieId NUMBER NOT NULL,
  pastaPubDate TIMESTAMP DEFAULT SYSTIMESTAMP
);
```

Script Output | Query Result | Query Result 1 | Query Result 2

Task completed in 0.037 seconds

```
Table PASTA created.
```

Insert data to pasta table

```
INSERT INTO pasta (pastaId, pastaName, pastaPrice, pastaDesc, pastaCategorieId, pastaPubDate) VA
(1, 'Spaghetti Carbonara', 99, 'A classic Italian pasta dish made with eggs, cheese, pancetta, a

INSERT INTO pasta (pastaId, pastaName, pastaPrice, pastaDesc, pastaCategorieId, pastaPubDate) VA
(2, 'Penne Arrabbiata', 149, 'A spicy pasta dish with penne pasta in a tomato sauce flavored wit

INSERT INTO pasta (pastaId, pastaName, pastaPrice, pastaDesc, pastaCategorieId, pastaPubDate) VA
(3, 'Fettuccine Alfredo', 159, 'A rich and creamy pasta dish with fettuccine noodles tossed in a

INSERT INTO pasta (pastaId, pastaName, pastaPrice, pastaDesc, pastaCategorieId, pastaPubDate) VA
(4, 'Spaghetti Bolognese', 139, 'A hearty pasta dish with spaghetti noodles topped with a slow-c

INSERT INTO pasta (pastaId, pastaName, pastaPrice, pastaDesc, pastaCategorieId, pastaPubDate) VA
(5, 'Pesto Pasta', 169, 'A vibrant pasta dish with spaghetti noodles tossed in a fresh basil pes

INSERT INTO pasta (pastaId, pastaName, pastaPrice, pastaDesc, pastaCategorieId, pastaPubDate) VA
(6, 'Vegetarian Lasagna', 129, 'A layered pasta dish with lasagna noodles, ricotta cheese, spina
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.705 seconds

```
1 row inserted.

1 row inserted.
```

Create user table

```
CREATE TABLE SITEDETAIL
(
  TEMPID NUMBER NOT NULL
, SYSTEMNAME VARCHAR2(21) NOT NULL
, EMAIL VARCHAR2(35) NOT NULL
, CONTACT1 NUMBER NOT NULL
, CONTACT2 NUMBER NOT NULL
, ADDRESS CLOB NOT NULL
);
```

Script Output | Query Result | Query Result 1 | Quer

Task completed in 0.048 seconds

```
Table SITEDETAIL created.
```

Insert data to user table

```
INSERT INTO sitedetail (tempId, systemName, email, contact1, contact2, address) VALUES
(1, 'Italy Pasta', 'pasta@gmail.com', '1234567890', '9876543210', 'Sample Address');
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

Task completed in 0.043 seconds

```
Table SITEDETAIL created.


1 row inserted.
```

Task three

Creating categories Table

```
test> // Categories Collection

test> db.createCollection("categories", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["categorieName"],
...           properties: {
...              categorieId: {
...                 bsonType: "int"
...              },
...              categorieName: {
...                 bsonType: "string",
...                 description: "must be a string and is required"
...              },
...              categorieDesc: {
...                 bsonType: "string"
...              },
...              categorieCreateDate: {
...                 bsonType: "date"
...              }
...           }
...        }
...     }
...  });
{ ok: 1 }
```

Creating contact Table

```
test>  // Contact Collection

test>  db.createCollection("contact", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["userId", "email", "phoneNo", "orderId", "message
", "time"],
...           properties: {
...              contactId: {
...                 bsonType: "int"
...              },
...              userId: {
...                 bsonType: "int"
...              },
...              email: {
...                 bsonType: "string"
...              },
...              phoneNo: {
...                 bsonType: "long"
...              },
...              orderId: {
...                 bsonType: "int"
...              },
...              message: {
...                 bsonType: "string"
...              },
...              time: {
...                 bsonType: "date"
...              }
...           }
...        }
...     }
...  });
```

Creating contactreply Table

```
test>  // ContactReply

test>  db.createCollection("contactreply", {
...      validator: {
...         $jsonSchema: {
...            bsonType: "object",
...            required: ["contactId", "userId", "message", "datetime"],
...            properties: {
...               id: {
...                  bsonType: "int"
...               },
...               contactId: {
...                  bsonType: "int"
...               },
...               userId: {
...                  bsonType: "int"
...               },
...               message: {
...                  bsonType: "string"
...               },
...               datetime: {
...                  bsonType: "date"
...               }
...            }
...         }
...      }
...   });
{ ok: 1 }
```

Creating DeliveryDetails Table

```
test> // DeliveryDetails

test>  db.createCollection("deliverydetails", {
...      validator: {
...         $jsonSchema: {
...            bsonType: "object",
...            required: ["orderId", "deliveryName", "deliveryPhoneNo", "de
liveryTime", "dateTime"],
...            properties: {
...               id: {
...                  bsonType: "int"
...               },
...               orderId: {
...                  bsonType: "int"
...               },
...               deliveryBoyName: {
...                  bsonType: "string"
...               },
...               deliveryBoyPhoneNo: {
...                  bsonType: "long"
...               },
...               deliveryTime: {
...                  bsonType: "int"
...               },
...               dateTime: {
...                  bsonType: "date"
...               }
...            }
...         }
...      }
...   });
```

Creating OrderItem Table

```
test> // OrderItems

test>  db.createCollection("orderitems", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["orderId", "pastaId", "itemQuantity"],
...           properties: {
...              id: {
...                 bsonType: "int"
...              },
...              orderId: {
...                 bsonType: "int"
...              },
...              ramenId: {
...                 bsonType: "int"
...              },
...              itemQuantity: {
...                 bsonType: "int"
...              }
...           }
...        }
...     }
...  });
{ ok: 1 }
```

Creating Order Table

```
test>  db.createCollection("orders", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["userId", "address", "zipCode", "phoneNo", "amou
Date"],
...           properties: {
...              orderId: {
...                 bsonType: "int"
...              },
...              userId: {
...                 bsonType: "int"
...              },
...              address: {
...                 bsonType: "string"
...              },
...              zipCode: {
...                 bsonType: "int"
...              },
...              phoneNo: {
...                 bsonType: "long"
...              },
...              amount: {
...                 bsonType: "int"
...              },
...              paymentMode: {
...                 enum: ["Cash", "Credit Card", "Debit Card", "Net
 Banking", "UPI"]
...              },
...              orderStatus: {
...                 enum: ["Pending", "Processing", "Delivered", "Ca
ncelled"]
...              },
...              orderDate: {
...                 bsonType: "date"
...              }
...           }
...        }
```

Creating pasta Table

```
db.createCollection("pasta", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["pastaName", "pastaPrice", "pastaDesc", "pastaCategorieI
d", "pastaPubDate"],
...           properties: {
...             pastaId: {
...                bsonType: "int"
...             },
...             pastaName: {
...                bsonType: "string"
...             },
...             pastaPrice: {
...                bsonType: "int"
...             },
...             pastaDesc: {
...                bsonType: "string"
...             },
...             pastaCategorieId: {
...                bsonType: "int"
...             },
...             pastaPubDate: {
...                bsonType: "date"
...             }
...           }
...        }
...     }
...  });
{ ok: 1 }
```

Creating sitedetail Table

```
test>  db.createCollection("sitedetail", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["systemName", "email", "contact1", "contact2", "addres
, "dateTime"],
...           properties: {
...              tempId: {
...                 bsonType: "int"
...              },
...              systemName: {
...                 bsonType: "string"
...              },
...              email: {
...                 bsonType: "string"
...              },
...              contact1: {
...                 bsonType: "long"
...              },
...              contact2: {
...                 bsonType: "long"
...              },
...              address: {
...                 bsonType: "string"
...              },
...              dateTime: {
...                 bsonType: "date"
...              }
...           }
...        }
...     }
...  });
{ ok: 1 }
```

Creating user Table

```
test>  db.createCollection("users", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["username", "firstName", "lastName", "email", "phone",
userType", "password", "joinDate"],
...           properties: {
...              id: {
...                 bsonType: "int"
...              },
...              username: {
...                 bsonType: "string"
...              },
...              firstName: {
...                 bsonType: "string"
...              },
...              lastName: {
...                 bsonType: "string"
...              },
...              email: {
...                 bsonType: "string"
...              },
...              phone: {
...                 bsonType: "long"
...              },
...              userType: {
...                 enum: ["Admin", "Customer"]
...              },
...              password: {
...                 bsonType: "string"
...              },
...              joinDate: {
...                 bsonType: "date"
...              }
...           }
...        }
...     }
...  });
```

Creating viewcart Table

```
test>  db.createCollection("viewcart", {
...     validator: {
...        $jsonSchema: {
...           bsonType: "object",
...           required: ["pastaId", "itemQuantity", "userId", "addedDate"],
...           properties: {
...              cartItemId: {
...                 bsonType: "int"
...              },
...              pastaId: {
...                 bsonType: "int"
...              },
...              itemQuantity: {
...                 bsonType: "int"
...              },
...              userId: {
...                 bsonType: "int"
...              },
...              addedDate: {
...                 bsonType: "date"
...              }
...           }
...        }
...     }
...  });
{ ok: 1 }
```

insert data into categories table

```
test> db.categories.insertMany([
...     {
...         categorieId: 1,
...         categorieName: "Spaghetti Bolognese",
...         categorieDesc: "Enjoy our classic Spaghetti Bolognese, featuring a
ch and savory meat sauce served over al dente spaghetti, topped with freshly gr
ed Parmesan cheese.",
...         categorieCreateDate: new Date("2021-03-17T18:16:28Z")
...     },
...     {
...         categorieId: 2,
...         categorieName: "Fettuccine Alfredo",
...         categorieDesc: "Indulge in our creamy Fettuccine Alfredo, made with
 rich Alfredo sauce and tender fettuccine noodles, garnished with parsley and g
ted Parmesan.",
...         categorieCreateDate: new Date("2021-03-17T18:17:14Z")
...     },
...     {
...         categorieId: 3,
...         categorieName: "Penne Arrabbiata",
...         categorieDesc: "Savor the spicy flavors of our Penne Arrabbiata, fe
uring penne pasta tossed in a fiery tomato sauce with garlic, red chili peppers
and olive oil.",
...         categorieCreateDate: new Date("2021-03-17T18:17:43Z")
...     },
...     {
...         categorieId: 4,
...         categorieName: "Lasagna",
...         categorieDesc: "Delight in our classic Lasagna, layered with rich r
t sauce, creamy béchamel, and tender pasta sheets, baked to perfection with a g
den cheese topping.",
...         categorieCreateDate: new Date("2021-03-17T18:19:10Z")
...     },
...     {
...         categorieId: 5,
...         categorieName: "Carbonara",
...         categorieDesc: "Experience the authentic taste of our Carbonara, ma
```

insert data into contact table

```
test> db.contact.insertOne({
...     contactId: 1,
...     userId: 2,
...     email: 'customer1@gmail.com',
...     phoneNo: NumberLong("1234567890"),
...     orderId: 1,
...     message: 'Nice You guys are doing great..',
...     time: new Date("2023-01-30T20:39:41Z")
... });
{
  acknowledged: true,
  insertedId: ObjectId('66473fc1cb38aa811946b7a1')
}
```

insert data into contactreply table

```
test> db.contactreply.insertMany([
...     {
...         id: 1,
...         contactId: 1,
...         userId: 2,
...         message: "Thank you for shopping with us",
...         datetime: new Date("2023-01-30T20:40:08Z")
...     }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6647411ccb38aa811946b7a2') }
}
test>
```

insert data into deliverydetails table

```
test> db.deliverydetails.insertMany([
...     {
...         id: 1,
...         orderId: 1,
...         deliveryName: 'John Doe',
...         deliveryPhoneNo: 9876543210,
...         deliveryTime: 30,
...         dateTime: new Date("2023-05-10T10:15:00Z")
...     },
...     {
...         id: 2,
...         orderId: 2,
...         deliveryName: 'Jane Smith',
...         deliveryPhoneNo: 8765432109,
...         deliveryTime: 35,
...         dateTime: new Date("2023-05-11T12:30:00Z")
...     },
...     {
...         id: 3,
...         orderId: 3,
...         deliveryName: 'Michael Johnson',
...         deliveryPhoneNo: 7654321098,
...         deliveryTime: 20,
...         dateTime: new Date("2023-05-12T14:45:00Z")
...     },
...     {
...         id: 4,
...         orderId: 4,
...         deliveryName: 'Emily Brown',
...         deliveryPhoneNo: 6543210987,
...         deliveryTime: 15,
...         dateTime: new Date("2023-05-13T16:00:00Z")
...     },
...     {
...         id: 5,
...         orderId: 8,
...         deliveryName: 'David Wilson',
```

insert data into orderitems table

```
test> db.orderitems.insertMany([
...     { id: 1, orderId: 1, pastaId: 1, itemQuantity: 2 },
...     { id: 2, orderId: 2, pastaId: 1, itemQuantity: 1 },
...     { id: 3, orderId: 2, pastaId: 2, itemQuantity: 1 },
...     { id: 4, orderId: 2, pastaId: 3, itemQuantity: 1 },
...     { id: 5, orderId: 3, pastaId: 1, itemQuantity: 1 },
...     { id: 6, orderId: 3, pastaId: 2, itemQuantity: 1 },
...     { id: 7, orderId: 3, pastaId: 3, itemQuantity: 1 },
...     { id: 8, orderId: 3, pastaId: 4, itemQuantity: 1 },
...     { id: 9, orderId: 3, pastaId: 5, itemQuantity: 1 },
...     { id: 10, orderId: 3, pastaId: 6, itemQuantity: 1 }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6647447ecb38aa811946b7b2'),
    '1': ObjectId('6647447ecb38aa811946b7b3'),
    '2': ObjectId('6647447ecb38aa811946b7b4'),
    '3': ObjectId('6647447ecb38aa811946b7b5'),
    '4': ObjectId('6647447ecb38aa811946b7b6'),
    '5': ObjectId('6647447ecb38aa811946b7b7'),
    '6': ObjectId('6647447ecb38aa811946b7b8'),
    '7': ObjectId('6647447ecb38aa811946b7b9'),
    '8': ObjectId('6647447ecb38aa811946b7ba'),
    '9': ObjectId('6647447ecb38aa811946b7bb')
  }
}
```

insert data into order table

```
test> // Inserting data into the "orders" collection

test>  db.orders.insertMany([
...       {
...           orderId: 1,
...           userId: 2,
...           address: 'London, UK',
...           zipCode: 111111,
...           phoneNo: NumberLong("2079461234"),
...           amount: 198,
...           paymentMode: 'Cash',
...           orderStatus: 'Pending',
...           orderDate: ISODate('2023-01-30T20:35:27.000Z')
...       },
...
...       {
...           orderId: 2,
...           userId: 2,
...           address: 'Manchester, UK',
...           zipCode: 111111,
...           phoneNo: 1611234567, // Adjusting phone number to integer
...           amount: 447,
...           paymentMode: 'Cash',
...           orderStatus: 'Pending',
...           orderDate: ISODate('2023-02-09T22:21:44.000Z')
...       },
...
...       {
...           orderId: 3,
...           userId: 2,
...           address: 'Birmingham, UK',
```

insert data into pasta table

```
test> db.pasta.insertMany([
...       {
...           pastaId: 1,
...           pastaName: 'Fettuccine Alfredo',
...           pastaPrice: 159,
...           pastaDesc: 'A creamy Alfredo sauce served with fettuccine past
tter, Parmesan cheese, and garlic.',
...           pastaCategorieId: 1,
...           pastaPubDate: new Date('2021-03-17T21:22:07Z')
...       },
...       {
...           pastaId: 2,
...           pastaName: 'Rigatoni Bolognese',
...           pastaPrice: 139,
...           pastaDesc: 'A hearty meat sauce made with ground beef, tomatoe
ions, carrots, and celery, served with rigatoni pasta.',
...           pastaCategorieId: 1,
...           pastaPubDate: new Date('2021-03-17T21:23:05Z')
...       },
...       {
...           pastaId: 3,
...           pastaName: 'Linguine Pesto',
...           pastaPrice: 169,
```

insert data into sitedetail table

```
test> db.sitedetail.insertMany([
...     {
...         tempId: 1,
...         systemName: 'Italy pasta',
...         email: 'italypasta@gmail.com',
...         contact1: NumberLong("1234567890"),
...         contact2: NumberLong("9876543210"),
...         address: 'Sample Address',
...         dateTime: new Date('2024-05-16T00:00:00Z')
...     }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66474b57cb38aa811946b7dd') }
}
```

insert data into users table

```
test> db.users.insertMany([
...     {
...       id: 1,
...       username: 'nyi',
...       firstName: 'zaw',
...       lastName: 'nyi',
...       email: 'nyizaw@gmail.com',
...       phone: NumberLong("9898989898"),
...       userType: 'Customer', // Assuming '1' corre
...       password: '$2y$10$AAfxRFOYbl7FdN17rN',
...       joinDate: new Date('2022-09-13T11:40:58Z')
...     },
...     {
...       id: 2,
...       username: 'nicky',
...       firstName: 'nicky',
...       lastName: '1',
...       email: 'nicky@gmail.com',
...       phone: NumberLong("1234567890"),
...       userType: 'Customer', // Assuming '0' corre
...       password: '$2y$10$2klu7oTnY3Yl.',
...       joinDate: new Date('2023-09-09T20:34:18Z')
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66474bf0cb38aa811946b7de'),
    '1': ObjectId('66474bf0cb38aa811946b7df')
  }
}
```

Task four

Implementing SQL

The well-structured arrangement of tables in the SQL implementation, such as "categories", "contact", "contact-reply", "delivery-details", "order-items", "orders", and "ramen", reflects the qualities of the objects. Each table represents a different entity, and the qualities of these entities are explained by the columns of these tables, which are comparable to the properties of objects in programming. The ability to establish associations between different entities via the use of primary keys—unique identifiers for each record—and foreign keys—links across tables—reflects the relational structure of SQL databases. Because of its ACID (Atomicity, Consistency, Isolation, Durability) properties, this structured design ensures data integrity and supports complicated queries, which makes it ideal for scenarios involving intricate transactional procedures.

Using MongoDB

MongoDB is a document-oriented NoSQL database that uses document collections to store data in a manner akin to JSON. Using object-oriented programming, this approach unifies the related data into a single document. It increases query efficiency for hierarchical data structures and reduces the requirement for joins. A versatile schema that holds a range of data types and structures can be included in each document for semi-structured data and quickly evolving data models. Because of its horizontal scalability architecture and prowess in managing massive volumes of unstructured or semi-structured data, MongoDB offers flexibility unhindered by preset schemas.

analytical and comparative work.

A number of factors illustrate the benefits and drawbacks of SQL and MongoDB. SQL databases are great for organizing structured data with explicit relationships and well-defined schemas. The strict structure of SQL guarantees data consistency and makes sophisticated join operations easier, making it a useful tool for applications that require advanced relational queries and transaction management (Elmasri and Navathe, 2015). However, this rigidity may be a drawback in scenarios where data models are dynamic or where scaling demands surpass vertical scalability capabilities.

In contrast, MongoDB offers significant benefits in terms of flexibility and scalability. Data structures' schema-free design makes it simple to make changes to them without causing a lot of disturbance, which is useful in agile development environments where needs change regularly. The capacity to incorporate relevant data into publications might enhance reading comprehension and optimize data access patterns for specific kinds of queries. However, because MongoDB does not have built-in support for multi-document ACID transactions, applications that need strong consistency guarantees across many documents can find it challenging to use.

Use of the Pasta Ordering System Application

The particular requirements of the application will determine whether to use MongoDB or SQL for a ramen ordering system. The structured SQL technique is preferred for managing well-defined entities where complicated queries and data consistency are crucial, such as user accounts, purchase histories, and inventory management. guarantees dependable transaction processing and accurate reporting, both of which are essential for operational effectiveness and business insight. Personalized meals, real-time updates, and customer interactions—all include dynamic data that is perfectly suited to MongoDB's document-oriented architecture. Because of its flexibility, new features or data points can be readily added to the schema, allowing for a more responsive and iterative development process. Because MongoDB is scalable, it can be used for applications that have high.

The decision between MongoDB and SQL for a ramen ordering system ultimately boils down to the specific needs of the application in terms of consistency, scalability, data format, and development flexibility. SQL is best suited for scenarios requiring tight data integrity and complex relational queries, whereas MongoDB is best suited for applications that demand scalable performance and flexible schema management. The choice must be consistent with the application's strategic goals, balancing immediate technology requirements with long-term scalability and maintainability concerns.

Tasks Five and Six

Query a: A join of three or more tables – you should consider various types of join in this query (e.g. inner join, left/right/full outer joins, etc.) and the query must include a restriction on the rows selected

SQL code:

```
//Question 1
SELECT u.username, o.orderId, oi.pastaId, p.pastaName
FROM users u
LEFT JOIN orders o ON u.id = o.userId
LEFT JOIN orderitems oi ON o.orderId = oi.orderId
INNER JOIN pasta p ON oi.pastaId = p.pastaId
WHERE o.orderDate >= TO_TIMESTAMP('2023-01-01', 'YYYY-MM-DD') AND p.pastaPrice < 200;
```

Output

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 40 in 0.039 seconds

	USERNAME	ORDERID	PASTAID	PASTANAME
1	san	1	1	Spaghetti Carbonara
2	san	2	1	Spaghetti Carbonara
3	san	3	1	Spaghetti Carbonara
4	san	5	1	Spaghetti Carbonara
5	san	9	1	Spaghetti Carbonara
6	san	10	1	Spaghetti Carbonara
7	san	12	1	Spaghetti Carbonara
8	san	16	1	Spaghetti Carbonara
9	san	2	2	Penne Arrabbiata
10	san	3	2	Penne Arrabbiata
11	san	9	2	Penne Arrabbiata
12	san	10	2	Penne Arrabbiata
13	san	12	2	Penne Arrabbiata
14	san	16	2	Penne Arrabbiata
15	san	2	3	Fettuccine Alfredo
16	san	3	3	Fettuccine Alfredo
17	san	9	3	Fettuccine Alfredo
18	san	16	3	Fettuccine Alfredo
19	san	3	4	Spaghetti Bolognese

MongoDB code:

```
test>  //Question 1

test>  db.users.aggregate([
...      { $lookup: { from: 'orders', localField: 'id', foreignField: 'userId', a
s: 'user_orders' } },
...      { $unwind: { path: '$user_orders', preserveNullAndEmptyArrays: true } },
...      { $lookup: { from: 'orderitems', localField: 'user_orders.orderId', fore
ignField: 'orderId', as: 'order_items' } },
...      { $unwind: { path: '$order_items', preserveNullAndEmptyArrays: true } },
...      { $lookup: { from: 'pasta', localField: 'order_items.pastaId', foreignFi
eld: 'pastaId', as: 'pasta_info' } },
...      { $unwind: { path: '$pasta_info', preserveNullAndEmptyArrays: true } },
...      { $match: {
...          'user_orders.orderDate': { $gte: ISODate('2023-02-01T00:00:00Z') },
...          'pasta_info.pastaPrice': { $gt: 100 }
...      } },
...      { $project: { _id: 0, username: 1, orderId: '$user_orders.orderId', past
aId: '$order_items.pastaId', pastaName: '$pasta_info.pastaName' } }
...  ]);
```

Output

```
  {
    username: 'nicky',
    orderId: 2,
    pastaId: 1,
    pastaName: 'Fettuccine Alfredo'
  },
  {
    username: 'nicky',
    orderId: 2,
    pastaId: 2,
    pastaName: 'Rigatoni Bolognese'
  },
  {
    username: 'nicky',
    orderId: 2,
    pastaId: 3,
    pastaName: 'Linguine Pesto'
  },
  {
    username: 'nicky',
    orderId: 3,
    pastaId: 1,
    pastaName: 'Fettuccine Alfredo'
  },
  {
    username: 'nicky',
    orderId: 3,
    pastaId: 2,
    pastaName: 'Rigatoni Bolognese'
  },
```

Query b: A query which uses one (or more) of the UNION, DIFFERENCE or INTERSECT operators.

SQL Code:

```
//Question 2
(SELECT orderId, address, 'Order' AS type FROM orders)
UNION
(SELECT contactId, email, 'Contact' AS type FROM contact);
```

Output:

Script Output × | Query Result × | Query Result 1 × | Query Result 2 ×

SQL | All Rows Fetched: 18 in 0.034 seconds

	ORDERID	ADDRESS	TYPE
1	1	nicky@gmail.com	Contact
2	3	Naples, Italy	Order
3	5	Palermo, Italy	Order
4	8	Florence, Italy	Order
5	10	Verona, Italy	Order
6	11	Pisa, Italy	Order
7	13	Bari, Italy	Order
8	15	Perugia, Italy	Order
9	16	Catania, Italy	Order
10	1	Rome, Italy	Order
11	2	Milan, Italy	Order
12	4	Turin, Italy	Order
13	6	Genoa, Italy	Order
14	7	Bologna, Italy	Order
15	9	Venice, Italy	Order
16	12	Siena, Italy	Order
17	14	Parma, Italy	Order
18	17	Trieste, Italy	Order

MongoDB code and output

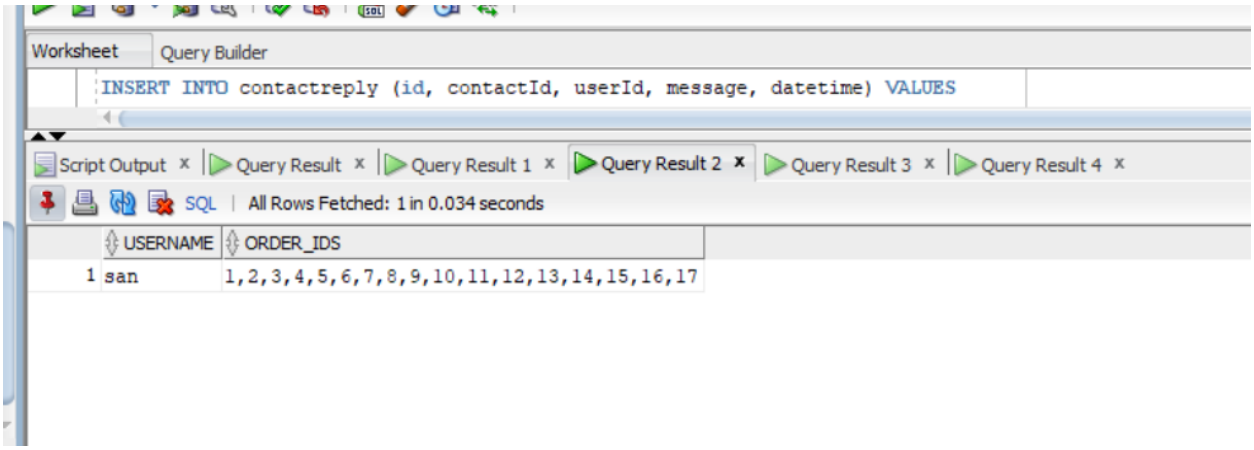
```
test>  db.orders.aggregate([
...      { $project: { _id: 0, orderId: 1, address: 1, type: 'Order' } },
...      { $unionWith: { coll: 'contact', pipeline: [
...          { $project: { _id: 0, contactId: '$contactId', email: '$email', type
: 'Contact' } }
...      ] } }
...  ]);
[
  { orderId: 1, address: 'London, UK', type: 'Order' },
  { orderId: 1, address: 'London, UK', type: 'Order' },
  { orderId: 2, address: 'Manchester, UK', type: 'Order' },
  { orderId: 3, address: 'Birmingham, UK', type: 'Order' },
  { orderId: 4, address: 'Leeds, UK', type: 'Order' },
  { orderId: 5, address: 'Glasgow, UK', type: 'Order' },
  { contactId: 1, email: 'customer1@gmail.com', type: 'Contact' }
]
```

Query c: A query which requires use of either a nested table or subtypes

SQL Code:

```
//Question 3
SELECT
    u.username,
    LISTAGG(o.orderId, ',') WITHIN GROUP (ORDER BY o.orderId) AS order_ids
FROM
    users u
JOIN
    orders o ON u.id = o.userId
GROUP BY
    u.username;
```

Output:



MongoDB code and output

```
test> //Question 3

test>  db.users.aggregate([
...       {
...           $lookup: {
...               from: "orders",
...               localField: "id",
...               foreignField: "userId",
...               as: "orders"
...           }
...       },
...       {
...           $project: {
...               _id: 0,
...               username: 1,
...               order_ids: "$orders.orderId"
...           }
...       }
...   ]);

[
  { username: 'admin', order_ids: [] },
  { username: 'nyi', order_ids: [] },
  { username: 'nicky', order_ids: [ 1, 1, 2, 3, 4, 5 ] }
```

Query d: A query using temporal features (e.g., timestamps, intervals, etc.) of Oracle SQL

SQL code:

```sql
//Question 4
SELECT
    EXTRACT(YEAR FROM orderDate) AS order_year,
    EXTRACT(MONTH FROM orderDate) AS order_month,
    COUNT(*) AS order_count
FROM
    orders
WHERE
    orderDate >= TIMESTAMP '2023-01-01 00:00:00'
    AND orderDate > TIMESTAMP '2023-04-01 00:00:00'
GROUP BY
    EXTRACT(YEAR FROM orderDate),
    EXTRACT(MONTH FROM orderDate)
ORDER BY
    order_year, order_month;
```

Output:

```sql
INSERT INTO contactreply (id, contactId, userId, message, datetime) VALUES
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Result 3 | Query Result 4

SQL | All Rows Fetched: 1 in 0.034 seconds

	ORDER_YEAR	ORDER_MONTH	ORDER_COUNT
1	2023	5	17

MongoDB code and output

```
test>  db.orders.aggregate([
...     {
...       $match: {
...         orderDate: {
...           $gte: ISODate("2023-01-01T00:00:00.000Z"),
...           $lt: ISODate("2023-04-01T00:00:00.000Z")
...         }
...       }
...     },
...     {
...       $project: {
...         order_year: { $year: "$orderDate" },
...         order_month: { $month: "$orderDate" }
...       }
...     },
...     {
...       $group: {
...         _id: { year: "$order_year", month: "$order_month" },
...         order_count: { $sum: 1 }
...       }
...     },
...     {
...       $project: {
...         _id: 0,
...         order_year: "$_id.year",
...         order_month: "$_id.month",
...         order_count: 1
...       }
...     },
...     {
...       $sort: { order_year: 1, order_month: 1 }
...     }
...   ])
[
  { order_count: 2, order_year: 2023, order_month: 1 },
  { order_count: 4, order_year: 2023, order_month: 2 }
]
```

Query e: A query using OLAP (e.g., ROLLUP, CUBE, PARTITION) features of Oracle SQL

SQL code:

```sql
SELECT
    CASE
        WHEN GROUPING(orderdate) = 1 THEN
            'Total'
        ELSE
            to_char(orderdate, 'YYYY-MM')
    END AS order_month,
    CASE
        WHEN GROUPING(categoriename) = 1 THEN
            'Total'
        ELSE
            categoriename
    END AS categorie_name,
    COUNT(o.orderid) AS total_orders,
    SUM(amount) AS total_amount
FROM
    orders o
JOIN orderitems oi ON o.orderid = oi.orderid
JOIN pasta p ON oi.pastaid = p.pastaid
JOIN categories c ON p.pastacategorieid = c.categorieid
GROUP BY
    ROLLUP(to_char(orderdate, 'YYYY-MM'), categoriename, orderdate)
ORDER BY
    order_month,
    categorie_name
FETCH FIRST 17 ROWS ONLY;
```

Ouput:

SQL | All Rows Fetched: 16 in 0.057 seconds

	ORDER_MONTH	CATEGORIE_NAME	TOTAL_ORDERS	TOTAL_AMOUNT
1	2023-05	Spaghetti Carbonara	3	2391
2	2023-05	Spaghetti Carbonara	3	2001
3	2023-05	Spaghetti Carbonara	3	1341
4	2023-05	Spaghetti Carbonara	1	198
5	2023-05	Spaghetti Carbonara	3	2091
6	2023-05	Spaghetti Carbonara	4	2784
7	2023-05	Spaghetti Carbonara	2	596
8	2023-05	Spaghetti Carbonara	4	4984
9	2023-05	Spaghetti Carbonara	2	596
10	2023-05	Spaghetti Carbonara	3	1341
11	2023-05	Spaghetti Carbonara	4	4384
12	2023-05	Spaghetti Carbonara	1	99
13	2023-05	Spaghetti Carbonara	7	10241
14	Total	Spaghetti Carbonara	40	33047
15	Total	Total	40	33047
16	Total	Total	40	33047

MongoDB code:

```
test> //Question 3

test>  db.orders.aggregate([
...      {
...          $lookup: {
...              from: "orderitems",
...              localField: "orderid",
...              foreignField: "orderid",
...              as: "order_items"
...          }
...      },
...      {
...          $unwind: "$order_items"
...      },
...      {
...          $lookup: {
...              from: "pasta",
...              localField: "order_items.pastaid",
...              foreignField: "pastaid",
...              as: "pasta"
...          }
...      },
...      {
...          $unwind: "$pasta"
...      },
...      {
...          $lookup: {
...              from: "categories",
...              localField: "pasta.pastacategorieid",
...              foreignField: "categorieid",
...              as: "category"
...          }
...      },
```

```
...         },
...         {
...             $unwind: "$category"
...         },
...         {
...             $group: {
...                 _id: {
...                     order_month: { $cond: [{ $eq: ["$orderdate", null] }, "
Total", { $dateToString: { format: "%Y-%m", date: "$orderdate" } }] },
...                     categorie_name: { $cond: [{ $eq: ["$category.categorien
ame", null] }, "Total", "$category.categoriename" ]}

...                 },
...                 total_orders: { $sum: 1 },
...                 total_amount: { $sum: "$order_items.amount" }
...             }
...         },
...         {
...             $sort: {
...                 "_id.order_month": 1,
...                 "_id.categorie_name": 1
...             }
...         },
...         {
...             $limit: 17
...         }
...     ]);
[ { _id: { order_month: null }, total_orders: 4800, total_amount: 0 } ]
test>

test>

test> db.orders.aggregate([
...     {
...         $lookup: {
...           from: "orderitems",
...           localField: "orderId",
```

```
..          as: "orderitems"
..        }
..      },
..      {
..        $unwind: "$orderitems"
..      },
..      {
..        $lookup: {
..          from: "pasta",
..          localField: "orderitems.pastaId",
..          foreignField: "pasta",
..          as: "tpasta"
..        }
..      },
..      {
..        $unwind: "$pasta"
..      },
..      {
..        $lookup: {
..          from: "categories",
..          localField: "pasta.tacoCategorieId",
..          foreignField: "categorieId",
..          as: "category"
..        }
..      },
..      {
..        $unwind: "$category"
..      },
..      {
..        $group: {
..          _id: {
..            order_month: {
..              $cond: {
..                if: { $eq: ["$orderDate", null] },
..                then: "Total",
..                else: { $dateToString: { format: "%Y-%m", date:
"$orderDate" } }
..              }
```

```
...            }
...          },
...          categorie_name: {
...            $cond: {
...              if: { $eq: ["$category.categorieName", null] },

...              then: "Total",
...              else: "$category.categorieName"
...            }
...          },
...          order_date: "$orderDate"
...        },
...        total_orders: { $sum: 1 },
...        total_amount: { $sum: "$amount" }
...      }
...    },
...    {
...      $sort: { "_id.order_month": 1, "_id.categorie_name": 1 }
...    },
...    {
...      $limit: 17
...    },
...    {
...      $project: {
...        order_month: "$_id.order_month",
...        categorie_name: "$_id.categorie_name",
...        total_orders: 1,
...        total_amount: 1,
...        _id: 0
...      }
...    }
...  ]);
```

Output

```
[
  {
    total_orders: 2,
    total_amount: 396,
    order_month: '2023-01',
    categorie_name: 'Spaghetti Bolognese'
  },
  {
    total_orders: 2,
    total_amount: 396,
    order_month: '2023-01',
    categorie_name: 'VEG PASTA'
  },
  {
    total_orders: 8,
    total_amount: 8656,
    order_month: '2023-02',
    categorie_name: 'Spaghetti Bolognese'
  },
  {
    total_orders: 8,
    total_amount: 8656,
    order_month: '2023-02',
    categorie_name: 'VEG PASTA'
  }
]
```

References

McGraw-Hill, March 2019. *Database System Concepts.* [Online] Available at: https://db-book.com/ [Accessed 13 5 2024].

Anon., May 2024. *User Interviews: ROI and Why They Chose.* [Online] Available at: https://dkdlnw2x4nft5.cloudfront.net/buyer-guide/prodview-pp445qepfdy34.pdf [Accessed 13 5 2024].

Chodorow, K., May 2013. *MongoDB: The Definitive Guide, 2nd Edition.* [Online] Available at: https://www.oreilly.com/library/view/mongodb-the-definitive/9781449344795/ [Accessed 13 5 2024].