Last Name: Li First Name: Jiaxin Student ID: 19683688

1. [10 pts] Take a look at the ETL script that you ran in the instructions (`InterchangeETL.sqlpp`) and have a look at some of the sample objects from each collection. Which of the resulting dataset(s) in AsterixDB can be classified as ***not*** being in 1NF based on the DDL that you've been given? Identify and list the 1NF and non-1NF datasets' name(s) and briefly explain your answer(s). (*Reminder*: 1NF = Flat atomic-valued relational data.)

Not 1NF:

1. Ad(having composite attribute "seller" and multi valued attribute "categories")
2. Item(have composite attribute 'seller')
3. User(having composite attribute "Name" & "Phone" and multi valued attribute "categories")

1NF:

1. Picture
2. Rating

2. [10 pts] Looking back at the E-R diagram from the Interchange conceptual design, compare and contrast the relational (PostgreSQL) version of the schema from your past SQL HW assignments with the AsterixDB "schema". Here by "schema" we are referring to the way that the objects in the five collections are structured. You will find that we have made some rather different design decisions here in the NoSQL database case. Briefly explain, after looking at the AsterixDB schema and after also exploring the data (e.g., by looking at the DDL statements in the script and running exploratory queries like "`SELECT VALUE u FROM Users u LIMIT 10;`") for **Users** and similarly for **Item**, how we have captured the information from each of the following E-R entities differently in AsterixDB and what the benefit(s) (hint: JOIN operations) of this new design probably are. Be sure to think about all of the "advanced E-R goodies" involved in the original Interchange ER design and how each of them are being handled here (in comparison to the previous relational treatment).

User:

1. In the relational version of the schema, multi valued attributes such as "phone" and "categories" and subtypes "Buyers" and "Sellers" are captured through creating separate datasets.
2. In AsterixDB schema, since multivalued attributes are allowed, all multivalued attributes are included in one schema. And subtypes "Buyer" and "Seller" information is captured through two boolean attributes, "Is_buyer" and "Is_seller" . Benefits: save four joins: 1) when querying for User information of Buyer. 2)when querying for User information of Sellers. 3) when querying for phone of a user. 4) when querying for categories of a user.

Item:

1. In the relational version of the schema, seller information is separated into different attributes ("seller_user_id" & "list_date").
2. In AsterxDB schema, seller info (user_id and list-date) are included in one field as composite attributes. Subgroups of Item "Good" and "Service" are captured through two boolean attributes "Is_good" and "Is_service". Benefits: save two joins: 1) when querying item related info for a good. 2) when querying item related info for a service.

3. [10 pts] Now that you have digested the differences above, write and ***test*** a pair of DDL statements (**CREATE TYPE** and **CREATE DATASET**) that could be used to create a typed version of the **User** dataset. You might call your new dataset **TypedUser.** Once you have done so, verify that it works by inserting a few records from the untyped version of **User** and then querying your new dataset to see them there. (You can look at the ETL script we gave you to see how you can use an **INSERT** … **SELECT** DML query pattern to do this, or you can manually insert a few records by following the examples in the little Don Chamberlin script files on the wiki.) Show your work - including your DDL and DML statements - below.

```
USE Interchange;
DROP DATASET TypedUser IF EXISTS;
DROP TYPE TypedUser IF EXISTS;
CREATE TYPE TypedUser AS {
      user_id: string,
      email: string,
      name: {first: string, last: string},
      joined_date: date,
      is_buyer: boolean,
      is_seller: boolean,
      categories: {{string}},
      phone:       {{{number: string, kind: string}}},
      address: {street: string, city: string, zipcode: int}
};
CREATE DATASET TypedUser(TypedUser) PRIMARY KEY user_id;
INSERT INTO TypedUser([
{"user_id": "U13",
 "email":"vjk231@gmail.com",
 "name": {"first":"Terek","last":"Cruise"},
 "joined_date": date("2008-04-26"),
 "is_buyer":boolean(0),
 "is_seller":boolean(1),
 "categories":{{"cookie", "food", "breakfast"}},
 "phone":{{{"number":"156-593-3212", "kind":"work"},{"number":"256-593-3212", "kind":"mobile"}}},
 "address": { "street": "201 Main St.", "city": "St. Louis, MO", "zipcode": int(63101)}},
{"user_id": "U15",
 "email":"acx231@gmail.com",
 "name": {"first":"Trsk","last":"Horito"},
 "joined_date": date("2008-06-26"),
 "is_buyer":boolean(1),
```

```
  "is_seller":boolean(0),
  "categories":{{"sports", "food","health"}},
  "phone":{{{"number":"156-593-3212", "kind":"work"},{"number":"256-593-3212", "kind":"mobile"}}},
  "address": { "street": "220 Main St.", "city": "St. Berky, MO", "zipcode": int(51101)}}
]);

SELECT * FROM TypedUser
```

4. [7 pts] List the emails, full names, and zip codes of users who are both a buyer and a seller on the platform and who are a resident of the state 'West Virginia'. Order the results using their zip codes in ascending order.

[5 pts] Query:

```
USE Interchange;

SELECT u.email, u.name, u.address.zip

FROM User u

WHERE u.is_buyer=TRUE AND u.is_seller=TRUE AND u.address.state="West Virginia"

ORDER BY u.address.zip ASC
```

[1 pts] Result:


```
[
    {
        "email": "garcia98650@aol.com",
        "name": {
            "first": "Jeffrey",
            "last": "Garcia"
        },
        "zip": 63735
    },
    {
        "email": "Jermaine_Ponce713@yahoo.com",
        "name": {
            "first": "Jermaine",
            "last": "Ponce"
        },
        "zip": 95474
    }
]
```

5. [8 pts] List the user_id, last name, and the quality rating that was given by buyers who rated a seller who was selling an item with the item_id 'G9WMY'. Order the list in descending order of the seller's quality rating.

[6 pts] Query:

```
USE Interchange;
SELECT VALUE{u.user_id, u.name.last, r.quality}
FROM User u, Ratings r, Item i
WHERE u.is_buyer=TRUE
        AND r.buyer_id=u.user_id
        AND r.seller_id=i.seller.user_id
        AND i.item_id= "G9WMY"
ORDER BY r.quality DESC
```

[2 pts] Result:


```
[
    {
        "user_id": "3R3CX",
        "last": "Castaneda",
        "quality": 3
    },
    {
        "user_id": "S86V6",
        "last": "Wheeler",
        "quality": 3
    },
    {
        "user_id": "53DXQ",
        "last": "Rodriguez",
        "quality": 1
    }
]
```

6. [10 pts] List the user name, email address, and categories of interest for users whose email address ends in ‘aol.com’ and all of whose categories of interest start with the letter ‘D’. Be careful not to include users who don’t have any categories of interest.

[8 pts] Query:

```
USE Interchange;
SELECT VALUE{u.name, u.email, u.categories}
FROM User u
WHERE u.email LIKE '%aol.com'
        AND (EVERY ca in u.categories SATISFIES ca LIKE 'D%' )
        AND array_count(u.categories)>=1
```

[2 pts] Result:


```
[
    {
        "name": {
            "first": "Alan",
            "last": "Vargas"
        },
        "email": "Vargas90@aol.com",
        "categories": [
            "Dairy, Eggs, & Cheese"
        ]
    }
]
```

7. [10 pts] List the user_ids, user names (in 'last, first' string format, e.g., 'Carey, Mike'), and mobile phone numbers (i.e., for phones of the kind 'MOBILE') for all of the sellers who listed an item on the Interchange platform on the date "2022-07-17". Name the fields in your output user_id, user_name, and mobile; order the results alphabetically by user name.

[8 pts] Query:

```
USE Interchange;
SELECT VALUE {"user_id": u.user_id, "user_name": concat(u.name.last,' , ',u.name.first),
                "mobile": (SELECT P.number FROM u.phone P WHERE P.kind='mobile')}
FROM User u, Item i
WHERE i.seller.user_id = u.user_id AND i.seller.list_date ='2022-07-17'
ORDER BY u.name.last ASC
```

[2 pts] Result:

Andrews , Jim	[]	46YCO
Beasley , Rita	[]	G558B
Becker , Deanna	[]	Q9QW3
Benjamin , Melissa	[]	CHM29
Chase , Danielle	[]	3QPJE
Cole , Lisa	[{ "number": "396-123-9381" }]	ZST93
Ellis , Lisa	[{ "number": "+1-466-789-3896x657" }]	H7FKP
Franklin , Jessica	[{ "number": "+1-064-660-6929x535" }]	F98VS
Hardy ,	[{ "number": "001-219-656-4140x069" }]	HQSNA
Hobbs , Aaron	[{ "number": "(368)921-4500x933" }]	Y26NY
Johnson , Daniel	[{ "number": "+1-075-479-1461" }]	U5R93
Jones , Kim	[]	EJ8AM
Knight , Thomas	[]	1VPD1
Lee , Jose	[]	3M1YI
Maddox , Stephen	[]	RYVDX
Martinez , Richard	[{ "number": "(246)724-8457x825" }]	GLVQG
Miller , Ryan	[{ "number": "6175201435" }]	8MZDC
Morris , Dakota	[]	0QEPM
Norton ,	[]	FUAY0
Reed , Riley	[{ "number": "852.667.9845" }]	UEAU3
Sanchez , Amanda	[]	JUDOH
Sanders , Marvin	[]	UADIT
Schaefer ,	[]	VKCWC
Simmons , Amanda	[]	8AOIB
Singleton , Monica	[{ "number": "001-233-211-8301x874" }]	4MSI6

Smith , Carlos	[]	6LV2R
Smith , Wanda	[]	Z6UIX
Thompson , Jake	[{ "number": "0525350074" }]	67EYU
Walls , Deanna	[]	9ZP9F
Washington ,	[{ "number": "(351)826-8868x69372" }]	XAY57
West , Taylor	[]	U98HN
White , Dawn	[]	IXU4J

8. [10 pts] Write the (arguably much more natural) NoSQL version of the now infamous query 3 from HW6: For all non-purchased services that had an ad placed by its seller, list the service's item_id, item_name, price, and category. Augment the result for each such service with a nested list of its (ad_id, ad_plan) pairs for its advertisements as well as a field containing the number (count) of pictures of the item that are stored in the Interchange database.

[8 pts] Query:

```
SELECT DISTINCT VALUE {
        "item_id": i.item_id,
        "item_name": i.name,
        "item_price": i.price,
        "category": i.category,
        "Ad": {"ad_id": a.ad_id, "ad_plan": a.plan},
        "pic": ARRAY_COUNT((SELECT * FROM Item i, Ad a, Picture p
                WHERE i.is_service= TRUE
                AND i.buyer IS MISSING
                AND a.item_id=i.item_id
                AND a.seller.user_id=i.seller.user_id
                AND p.item_id=i.item_id))
        }
FROM User u, Item i, Ad a
WHERE i.is_service= TRUE
        AND i.buyer IS MISSING
        AND a.item_id=i.item_id
        AND a.seller.user_id=i.seller.user_id
```

[2 pts] Result:


```
[
    {
        "Ad": {
            "ad_id": "S9L79",
            "ad_plan": "platinum"
        },
        "pic": 2,
        "item_id": "8J5CL",
        "item_name": "Necklace",
        "item_price": 640.69,
        "category": "Clothing, Shoes & Jewelry"
    }
]
```

9. [10 pts] We would like to perform analytics on items, more specifically through aggregations of service categories and frequencies. Use the **ROLLUP** operator to report on the total number of items for each (category, frequency) combination as well as the subtotal for each category and the overall total as well. Order your results on the combination (category, frequency) in ascending order and use the UI's TABLE output format for this query. After examining the full output so that you get a full sense of what **ROLLUP** does, take a screenshot to show the result with 'Items per page' set to 10 and paste the query and the screenshot below.

[7 pts] Query:

```
USE Interchange;
SELECT i.category, i.frequency, count(i.item_id)as Total_count
FROM Item i
WHERE i.is_service = TRUE
GROUP BY ROLLUP(i.category, i.frequency) ORDER BY i.category, i.frequency ASC
```

[3 pts] Result:

Total_count	category	frequency
4974		
510	Arts, Crafts & Sewing	
88	Arts, Crafts & Sewing	daily
73	Arts, Crafts & Sewing	monthly
90	Arts, Crafts & Sewing	once
94	Arts, Crafts & Sewing	quarterly
71	Arts, Crafts & Sewing	weekly
94	Arts, Crafts & Sewing	yearly
504	Beauty & Personal Care	
92	Beauty & Personal Care	daily

10. [15 pts] One last additional analytical query has come to mind – the Interchange CEO wants to analyze item sales, and they would like to do so by category! For the items that have been sold, report on their item_id, their price, their date of sale, category, and their rank within their category (where the highest-priced item in a category would be ranked 1, the next highest-priced item in the same category 2, and so on, with each category having its own ranked order). Use SQL++'s window-based aggregation (i.e., the **OVER** clause) to find the ranks. Limit the per-result rankings to the top three items in each category. **Note**: do not use LIMIT here, because if there happens to be a price tie, the CEO still wants to see results for each of the categories' top three prices – which would require more than three results to be shown for a category with such a tie.

In order to fully solve this problem and demonstrate the correctness of your query, you should do the following: (a) Write and debug the query. (b) Examine the results. (c) Perform a SQL++ **UPSERT** operation to make the prices of the first two objects in the now-working query's results the same (set them to the higher of the two). (d) Re-run the query to show that it indeed handles price ties in the way that the CEO wants. Show your **OVER** query, **UPSERT** statement, and the final query output (i.e., from step (d)) below. Show your output using the same approach prescribed for problem 9.

[8 pts] Query:

```
USE Interchange;
SELECT full_rank.p_rank, full_rank.item_id, full_rank.price, full_rank.purchase_date, full_rank.category
FROM (SELECT i.item_id, i.price, i.buyer.purchase_date, i.category, RANK() OVER (PARTITION BY i.category ORDER BY i.price DESC) as p_rank FROM Item i WHERE i.buyer.purchase_date IS NOT UNKNOWN) full_rank
WHERE full_rank.p_rank<=3
```

[5 pts] Upsert:

```
UPSERT INTO Item
("item",{
 "_oid": uuid("520e82e9-3cfc-56d4-4de3-ece049b879db"),
 "buyer":{"user_id":"PZG0D",
                    "purchase_date":"2022-10-02"},
"category":"Arts, Crafts & Sewing",
"description":null,
"frequency":"quarterly",
"item_id":"OD56O",
"name":"Paint Brushes",
 "price":float(1985.27),
 "seller":{"user_id":"GPYLB",
                                "list_date":"2022-01-19"},
          "is_good":false,
          "is_service":true})
```

[2 pts] Result from (d):

summary

{ "ranking": 1, "item_id": "OD56O", "price": 1985.27, "purchase_date": "2022-10-02", "category": "Arts, Crafts & Sewing" }

{ "ranking": 2, "item_id": "XAOZP", "price": 1985.27, "purchase_date": "2022-08-30", "category": "Arts, Crafts & Sewing" }

{ "ranking": 3, "item_id": "Y46O9", "price": 1946.79, "purchase_date": "2022-08-03", "category": "Arts, Crafts & Sewing" }

{ "ranking": 1, "item_id": "VQO3R", "price": 1994.19, "purchase_date": "2022-04-08", "category": "Beauty & Personal Care" }

{ "ranking": 2, "item_id": "B8LZY", "price": 1989.87, "purchase_date": "2022-10-18", "category": "Beauty & Personal Care" }

{ "ranking": 3, "item_id": "D4CC0", "price": 1979.11, "purchase_date": "2022-09-22", "category": "Beauty & Personal Care" }

{ "ranking": 1, "item_id": "8DFT7", "price": 1977.76, "purchase_date": "2022-10-17", "category": "Clothing, Shoes & Jewelry" }

{ "ranking": 2, "item_id": "BADTV", "price": 1968.79, "purchase_date": "2022-09-04", "category": "Clothing, Shoes & Jewelry" }

{ "ranking": 3, "item_id": "756GQ", "price": 1939.42, "purchase_date": "2022-10-29", "category": "Clothing, Shoes & Jewelry" }

{ "ranking": 1, "item_id": "6MR70", "price": 1977.75, "purchase_date": "2022-08-23", "category": "Electronics" }