Normalization 经典例题

Case: Veterinarian Clinic

- The table below is part of the medical records for a Veterinarian Clinic
- PRACTICE (Animal_ID, Animal, Animal type, OWNER_ID, Owner, Phone, Consult, PROC_ID, PROC_DESC)
- The combination of ANIMAL_ID and PROC_ID is the candidate key for the relation. The following functional dependencies hold:
  - Animal_ID → ANIMAL, ANIMAL Type, OWNER_ID
  - OWNER_ID → OWNER, PHONE
  - PROCID → DESCRIPTION
  - ANIMAL_ID, PROC_ID → Consult
- Please normalise the data to third normal form (3NF). Be sure to identify all anomalies, and show each stage (1NF, 2NF, 3NF).
- (Key: **BOLD** primary key *ITALIC* foreign key ***BOLD*** + *ITALIC* primary foreign key)

Case: Veterinarian Clinic

Animal ID	Animal	Animal Type	Owner_ID	Owner	Phone	Consult	ProcID	Description
317	Ralph	Dog	10	Julie Sumner	0409 673-888	13-Oct-13	101	Annual Checkup
317	Ralph	Dog	10	Julie Sumner	0409 673-888	27-Apr-13	115	Teeth Clean
317	Ralph	Dog	10	Julie Sumner	0409 673-888	14-Oct-14	119	3 month Checkup
398	Zeno	Canary	23	Tony Rijks	0408 322-444	21-Jul-14	105	Parasite treatment
398	Zeno	Canary	23	Tony Rijks	0408 322-444	14-Oct-13	119	3 month Checkup
441	Panda	Short haired cat	47	Helene Hanff	0419 121-212	24-Apr-14	715	Initial Consultation
441	Panda	Short haired cat	47	Helene Hanff	0419 121-212	27-Apr-13	115	Teeth Clean
518	Zeno	Canary	23	Tony Rijks	0408 322-444	1-Mar-15	001	6 month Checkup

Case: Veterinarian Clinic

Cas

1NF All atomic cells
However Insert Update Delete Anomalies exist
- DELETE Zeno the Canary we lose the 6 month checkup procedure (001)
- UPDATE Ralph's owner – multiple updates
- INSERT Have to know the consultation before we can add owner and pet

1NF
PRACTICE(**animalID**, animal, animal_type, owner_ID, owner, phone, **procID**, procedure, consult)
2NF
To be in 2NF - 1NF and No partial functional dependencies
Partial functional dependency exists in PRACTICE
animalID ➔ animal, animal type, owner_ID
ProcID ➔ procedure

ANIMAL(**animalid**, animal, animal type, owner_ID, owner, phone
PROCEDURE(**procid**, procedure)
PRACTICE2(***procID***, ***animalID***, consult)
Anomolies
If the owner changes we have multiple updates
3NF
To be in 3NF - 2NF and no transitive functional dependencies
Transitive functional dependencies exist
Owner_ID --> owner, phone not dependent on animal_ID of ANIMAL

PROCEDURE(**procid**, procedure)
PRACTICE2(***procID***, ***animalID***, consult)
ANIMAL2(**animal_id**, animal, animal_type, *owner_ID*)
OWNER(**owner_ID**, owner, phone)

Case: Office Inventory

- The table shown below is part of an office inventory database. Identify the design problems and draw a revised table structure in 3rd Normal Form (3NF) that corrects those problems. For each step explicitly identify and discuss which normal form is violated.
- (Key: PK = Bold FK = Italic PFK = Bold + Italic)
- Inventory (**ItemID**, Description, Qty, Cost/Unit, Dept, Dept Name, Dept Head)
- ItemID is the candidate key for this table.
- The following functional dependencies hold:
- Dept ⟶ Dept Name and Dept Head
- Qty, Cost/Unit ⟶ Inventory Value

ItemID	Description	Dept	Dept Name	Dept Head	Qty	Cost/Unit	Inventory Value
4011	1.4m Desk	MK	Marketing	Jane Thompson	5	200	1000
4020	Filing Cabinet	MK	Marketing	Jane Thompson	10	75	750
4005	Executive chair	MK	Marketing	Jane Thompson	5	100	500
4036	1.2m Desk	ENG	Engineering	Ahmad Rashere	7	200	1400

Case: Office Inventory

Case: Office Inventory

1NF: All data is atomic – there are no repeating groups.
2NF: The table is in 1NF and there are no partial functional dependencies, so the table is in 2NF.
3NF: The table is in 2NF however there is a transitive functional dependency:
Dept → Dept Name, Dept Head is not a key attribute. This transitive FD violates 3NF.

Quan, Cost/Unit → Inventory Value
Because there is derived value between these columns (Quan × Cost/Unit = Inventory Value), this FD can be resolved by removing the redundant Inventory Value column.

3NF
Inventory2(**Item ID**, Description, *Dept*, Quan, Cost/Unit)
Department (**Dept**, Dept Name, Dept Head)

KEY:
BOLD = PK
ITALIC = FK
BOLD + ITALIC = PFK

Case: Food Delivery

- We are modelling a database for store data about food delivery. For each delivery, we must record its id number and the restaurant which is selected to provide the take-away food, including the restaurant’ s id, name, phone number and address. Each delivery must be completed by one deliveryman. We must record the deliveryman’ s id, name and the selected payment method. In addition, we must record the food ordered, including its id, name and price.
- Our modeller has arrived at the following relation. But this relation is not in third normal form.
- DELIVERY (deliveryId, restaurantId, restaurantName, restaurantPhone, restaurantAddress, deliverymanId, deliverymanName, paymentMethod, (foodId, foodName, foodPrice)) Your job is to convert the relation to 3rd normal form.
- Mark your primary keys with a solid underline, and your foreign keys with a dotted underline. (Any attributes that are both primary and foreign keys should get both underlines.)
- You don’ t need to show intermediate normal forms – just the 3rd normal form you end up with.

Case: Food Delivery

Case: Food Delivery

Delivery(deliveryId, restaurantId, deliverymanId, foodId, paymentMethod)

Restaurant(restaurantId, restaurantName, restaurantPhone, restaurantAddress,)

Deliveryman(deliverymanId, deliverymanName)

Food(foodId, foodName, foodPrice)

Case: Students and Marks

We are modelling a database for storing students and their marks. Students have an id number and name, and receive marks for a series of assignments, each of which has in id number and a title. A student does a given assignment only once – we record the date that the student submitted the assignment.

Our modeller has arrived at the following relation (primary key is underlined).

STUDENTMARKS (studentid, givenName, surname (assignmentId, assignmentTitle, mark, dateSubmitted))

But this relation is not in third normal form.

First, explain why it is not. (1 mark)

Second, convert the relation to 3rd normal form.
Mark your primary keys with a solid underline, and your foreign keys with a dotted underline.

You don't need to show intermediate normal forms – just the 3rd normal form you end up with. (4 marks)

Case: Students and Marks

Case: Students and Marks

STUDENTMARKS(studentid, givenName, surname, (assignmentId, assignmentTitle, mark, dateSubmitted))

1NF

STUDENTMARKS(studentid, givenName, surname)
SUBMISSION(*studentid*, assignmentId, assignmentTitle, mark, dateSubmitted)

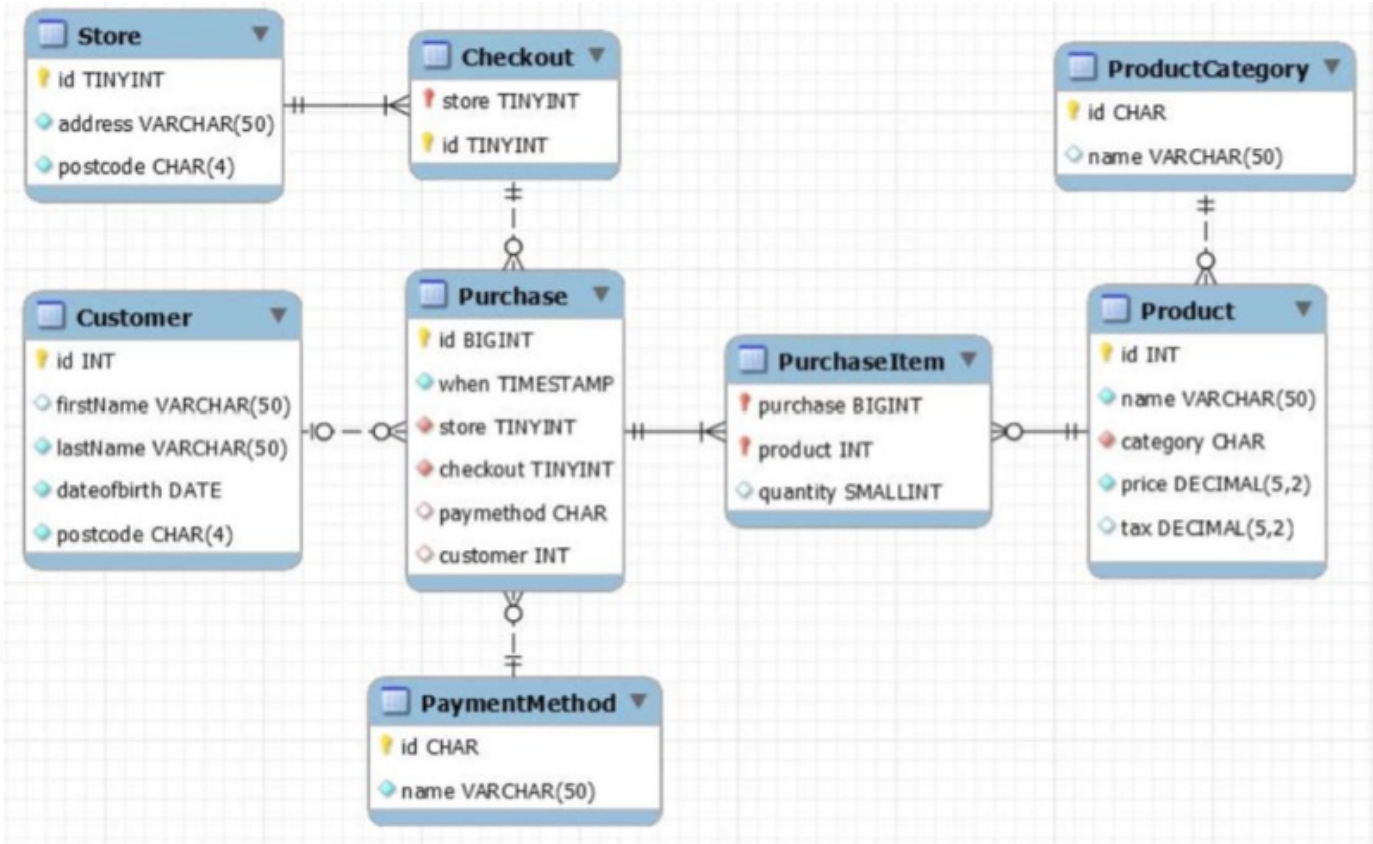
2NF

STUDENTMARKS(studentid, givenName, surname)
SUBMISSION(*studentid*, *assignmentId*, mark, dateSubmitted)
ASSIGNMENT(assignmentId, assignmentTitle)

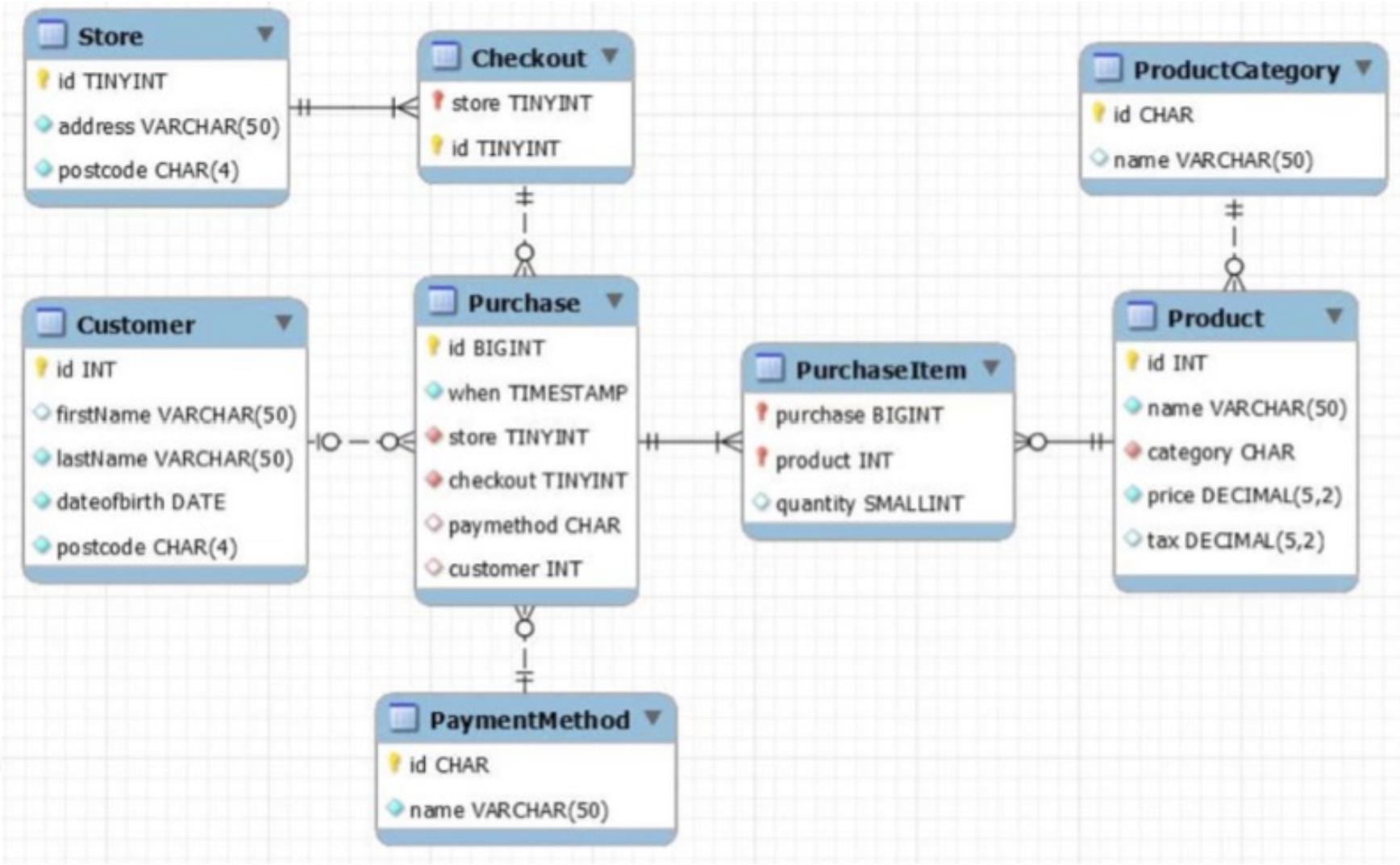
SQL 经典例题

Case: Checkout System

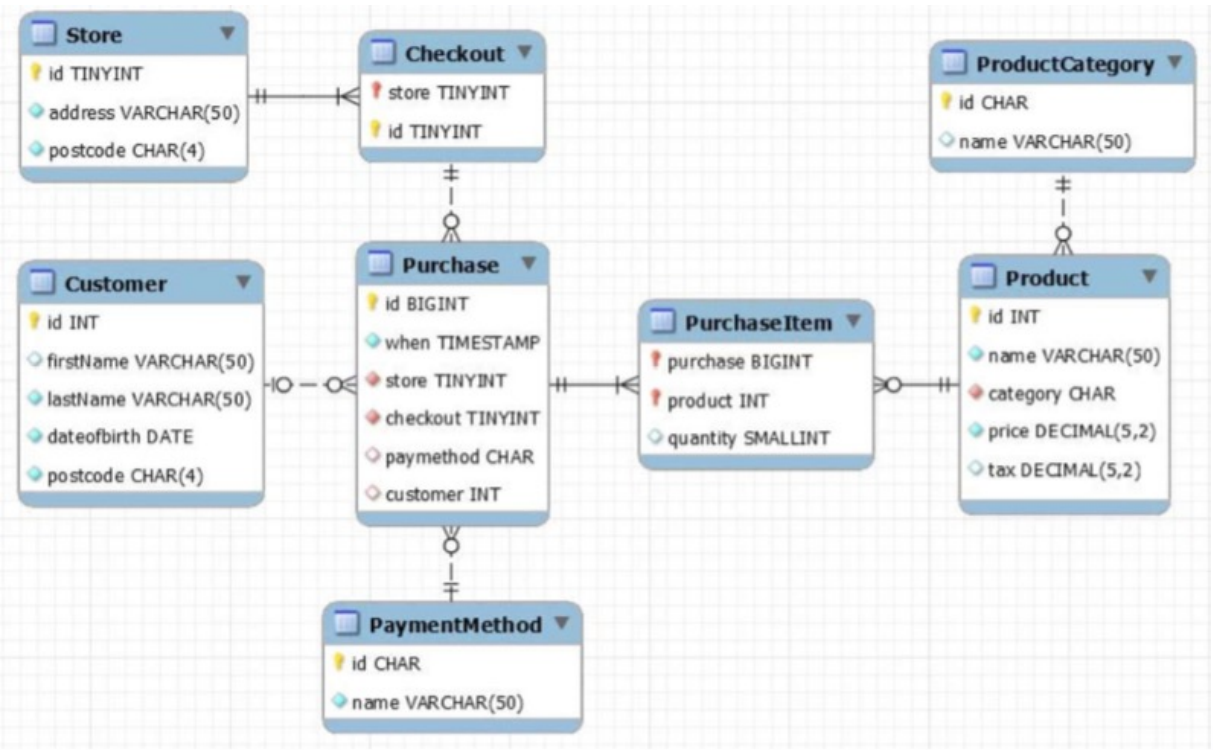
- Consider the following data model and sample data (not all data is shown) for a checkout system. Your job is to write SQL queries that answer questions posed by management. Our customers take a collection of products to a checkout for scanning: this collection is one "Purchase". Each item within the purchase is a "Purchase Item". Some customers identify themselves at the checkout by scanning a loyalty card.
- The checkouts are numbered within each store: a given store n has checkouts n-1, n-2 etc. In a given store, checkout 1 is the 'first' checkout; the checkout with the highest number is the 'last'. Each product is classified within a particular category. There are several payment methods.
- The following ER diagram describes the database schema which has been implemented.



Case: Checkout System

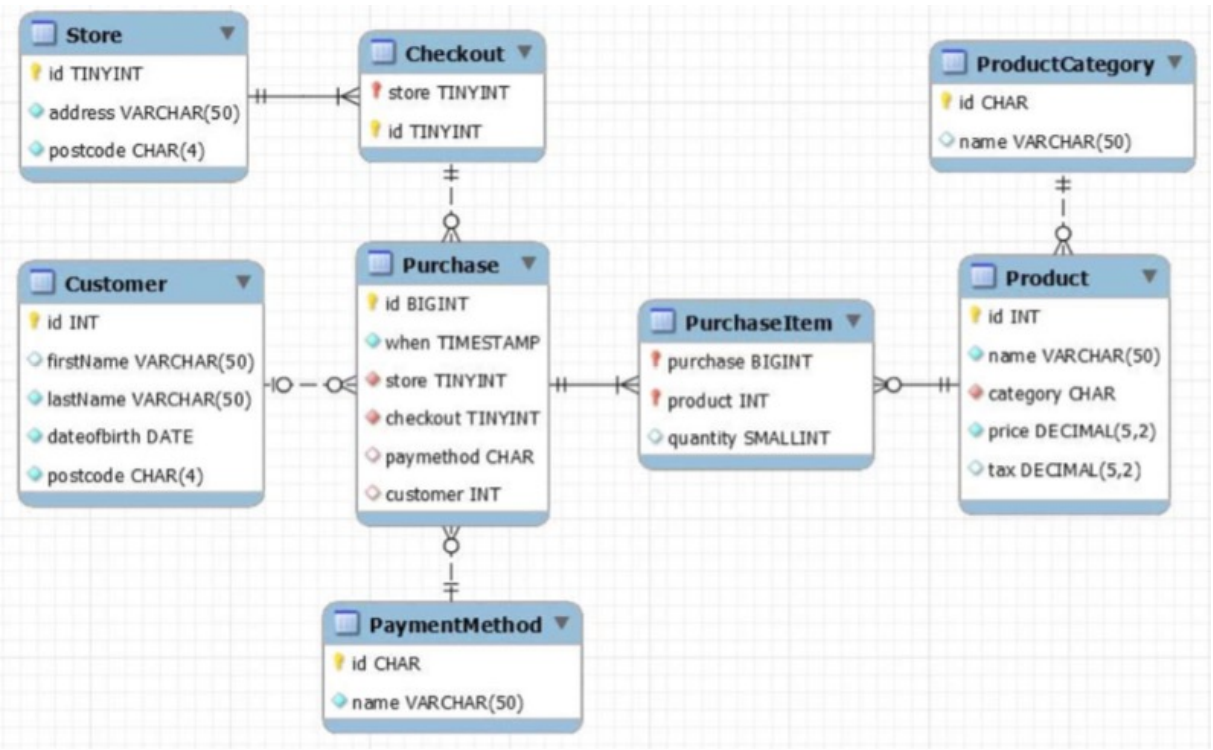


Case: Checkout System



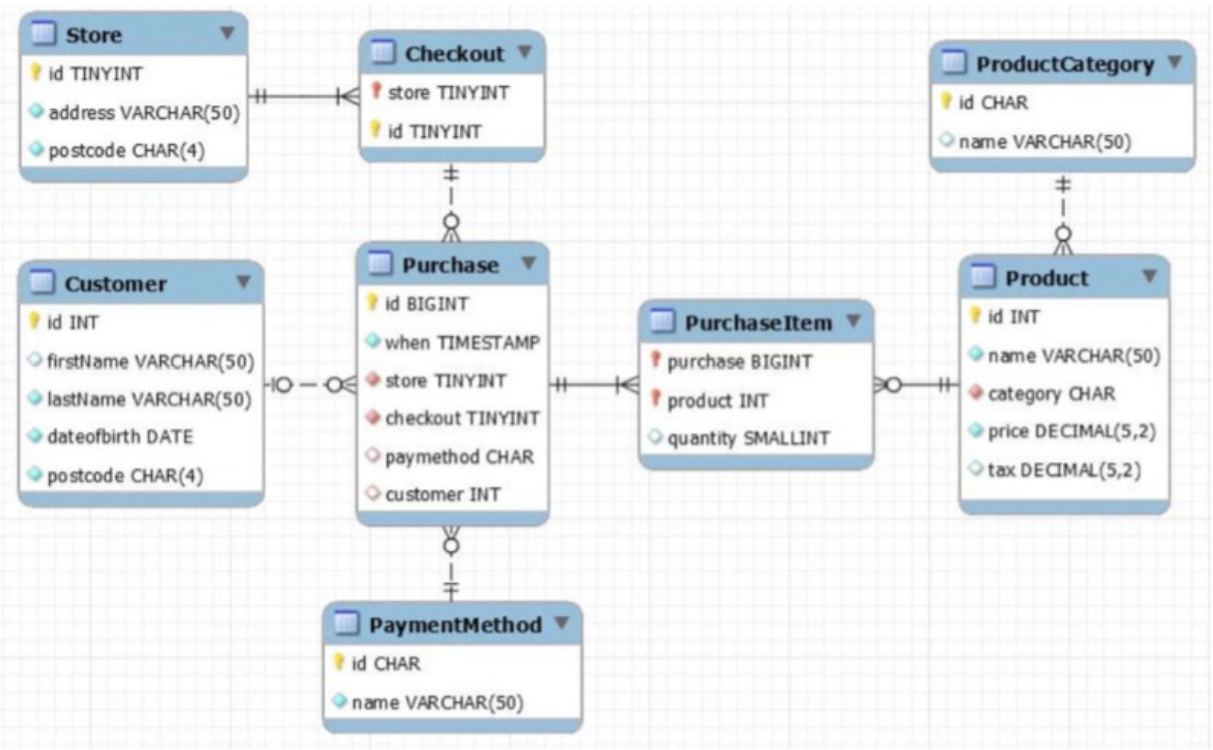
- 1. Which is the longest customer lastname that contains the letter 'E' ?

Case: Checkout System



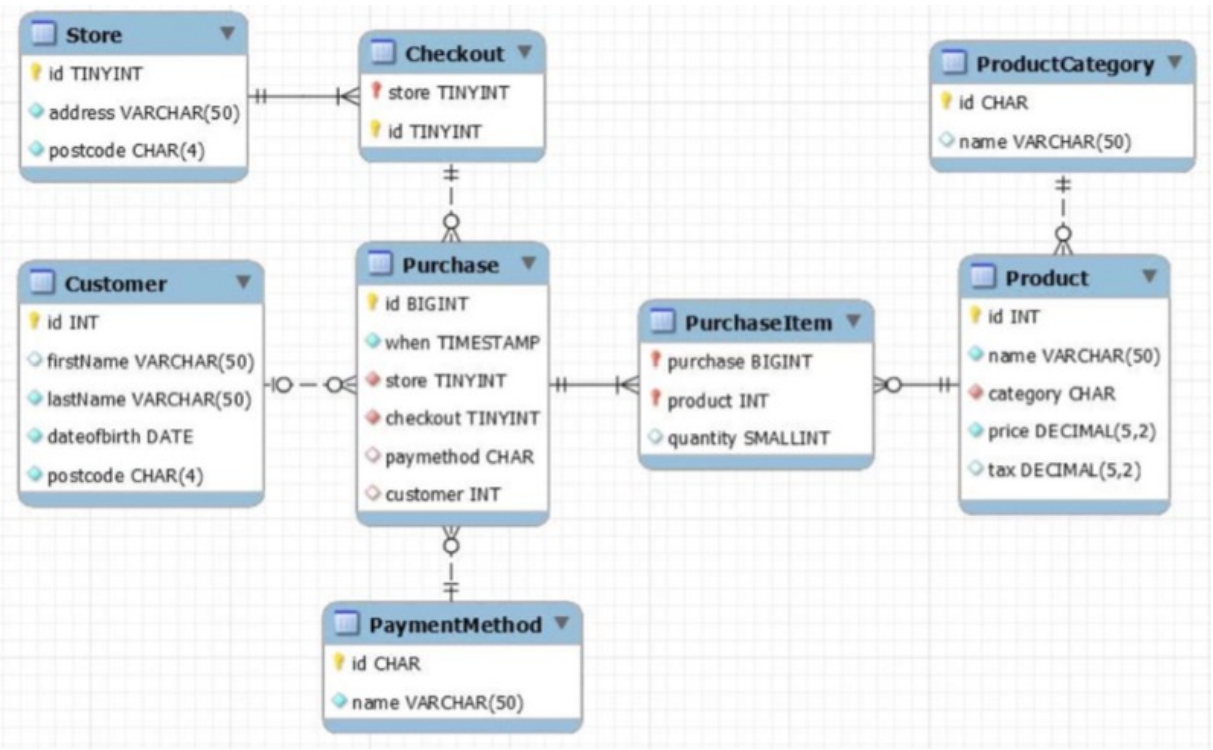
2. List the names and prices of all products of type ‘food’ that cost more than $3. Order them by descending order of price.

Case: Checkout System



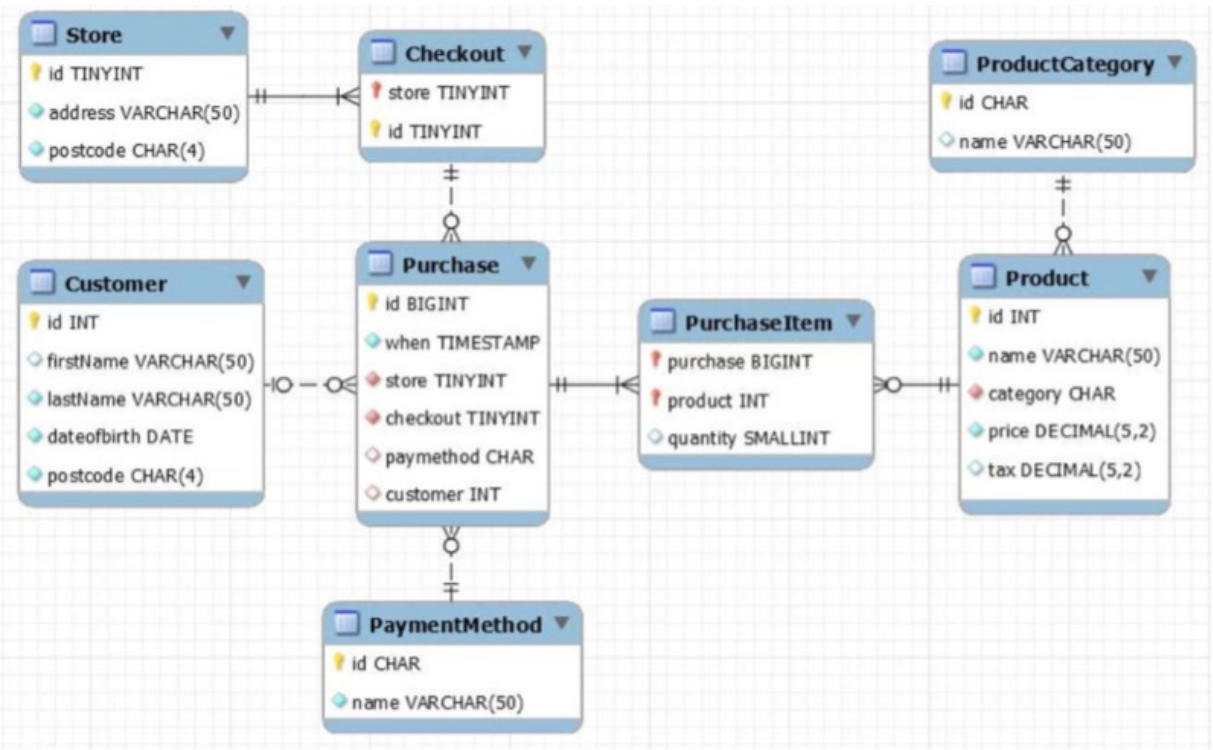
3. What is the average number of items that purchases contain?

Case: Checkout System



4. List the names of any customers who have used all payment methods.

Case: Checkout System



5. How many purchases were made at the "last" checkout in a store?

ANSWER - Case: Checkout System

1.
```
SELECT lastname FROM Customer
WHERE instr(lastname,'e') > 0
ORDER BY length(lastname) DESC
LIMIT 1;
```

2.
```
SELECT name, price
FROM product
WHERE price > 3 AND category in
    (SELECT id FROM productcategory
    WHERE name = "food")
ORDER BY price DESC;
```

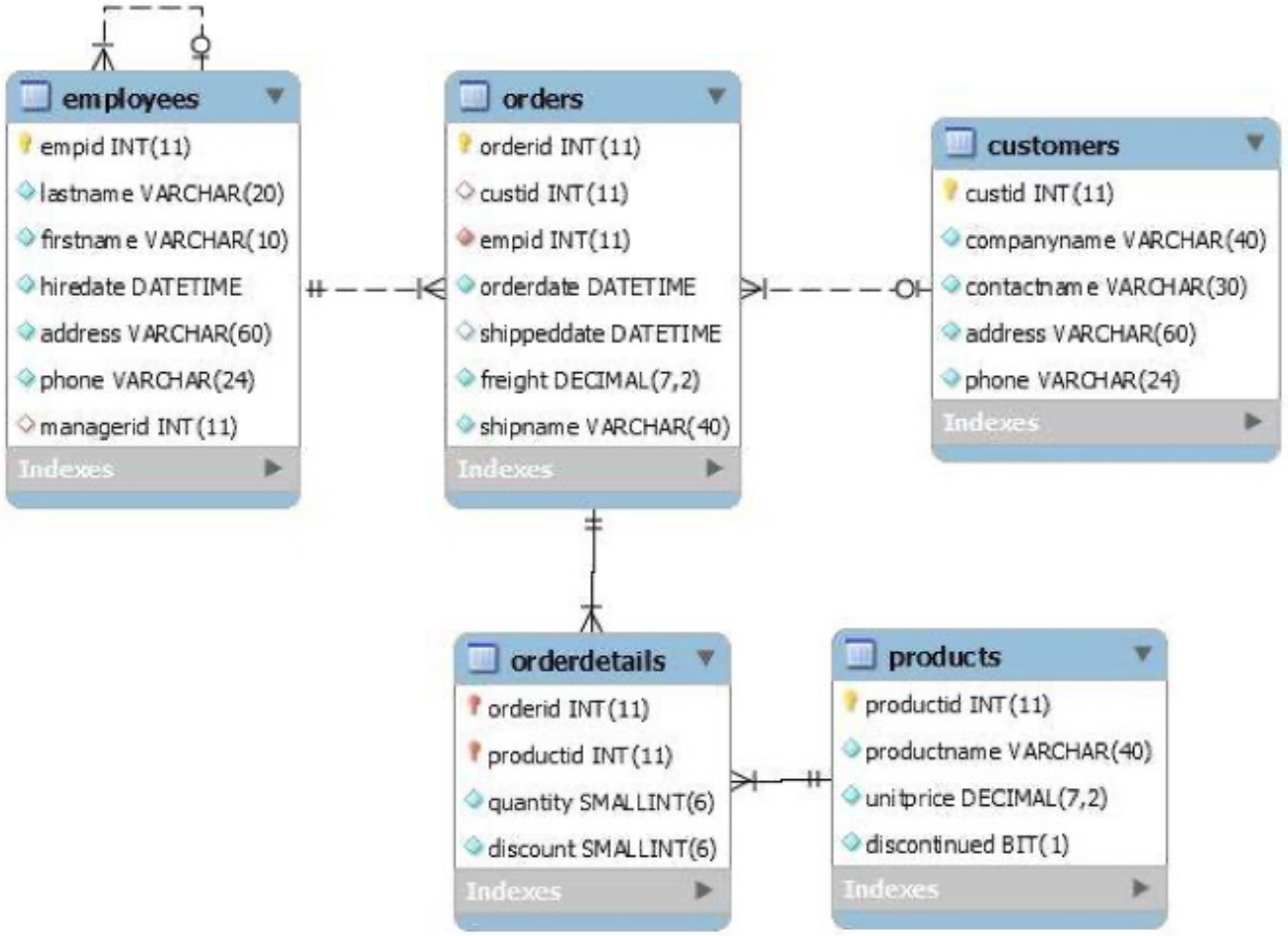
3.
```
SELECT ROUND(AVG(numItems),2) AS AvgItems
FROM (
    SELECT purchase, count(*) AS numItems
    FROM PurchaseItem
    GROUP BY purchase
) AS PurchaseCounts;
```

4.
```
SELECT firstname, lastname, COUNT(DISTINCT(paymethod)) AS numMethods
FROM Customer INNER JOIN Purchase ON Customer.id = Purchase.customer
GROUP BY Customer.id
HAVING numMethods =
    (SELECT COUNT(*) FROM PaymentMethod);
```

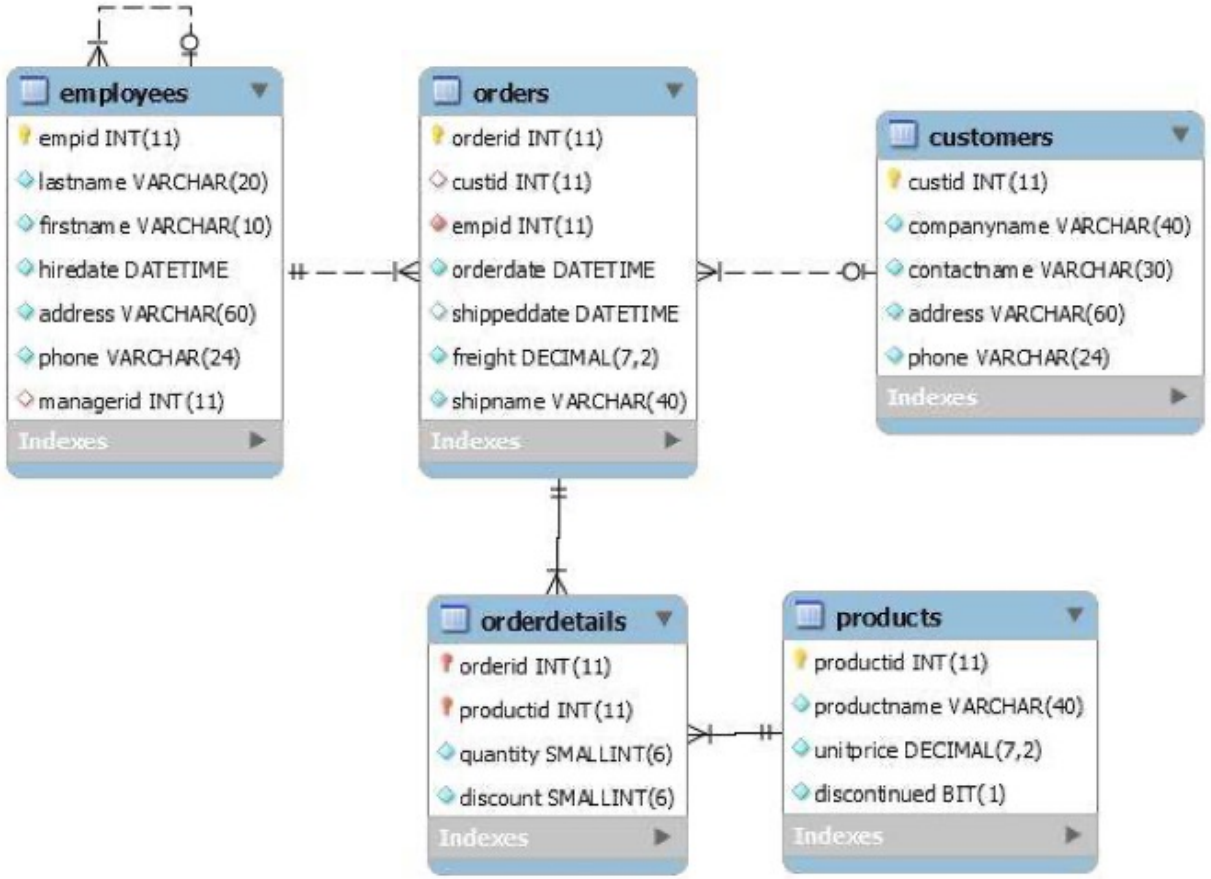
5.
```
SELECT COUNT(*) as TotalPurchases
FROM Purchase
WHERE checkout =
    (SELECT MAX(id) FROM Checkout
    WHERE Checkout.store = Purchase.store);
```

Case: Store System

- Given the schema in Figure 2, write a single SQL statement to correctly answer each of the following questions (3A – 3D). DO NOT USE VIEWS to answer questions.

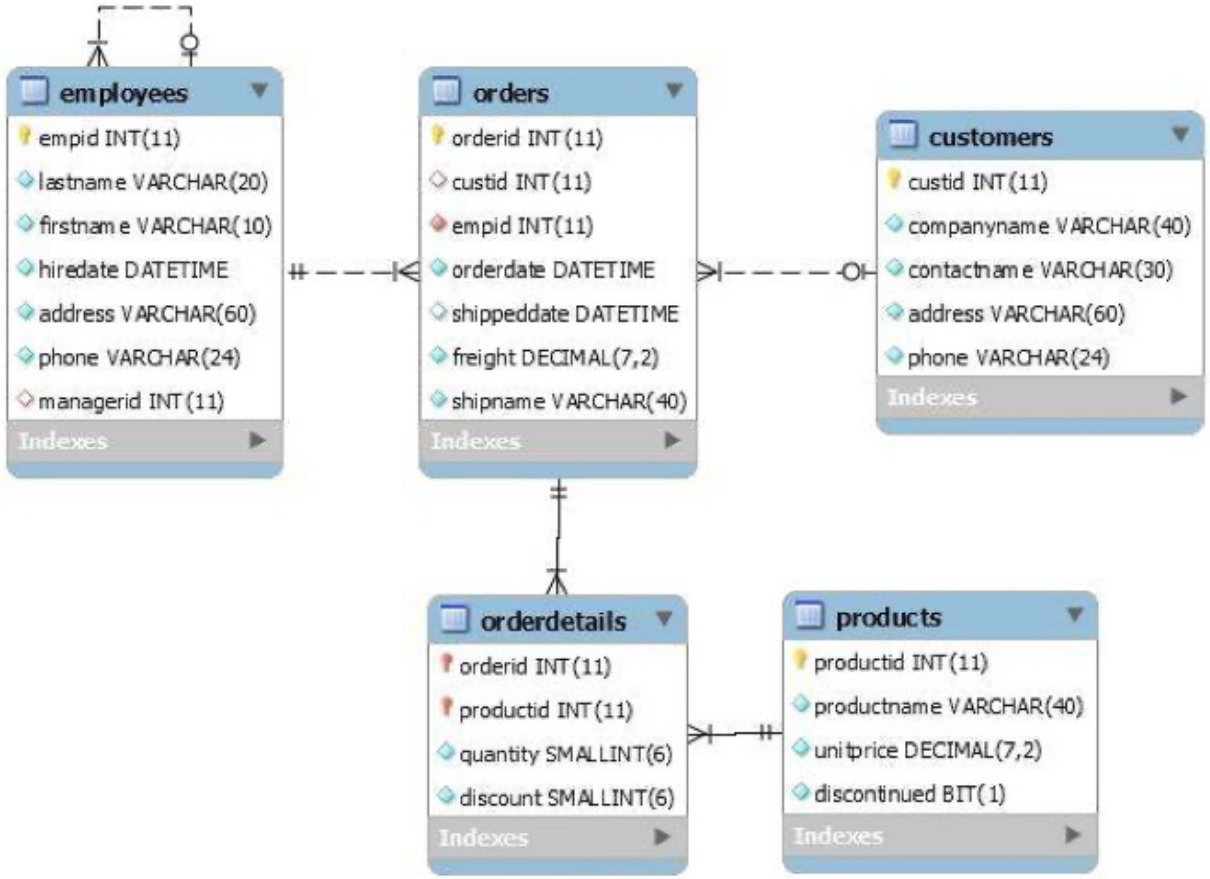


Case: Store System



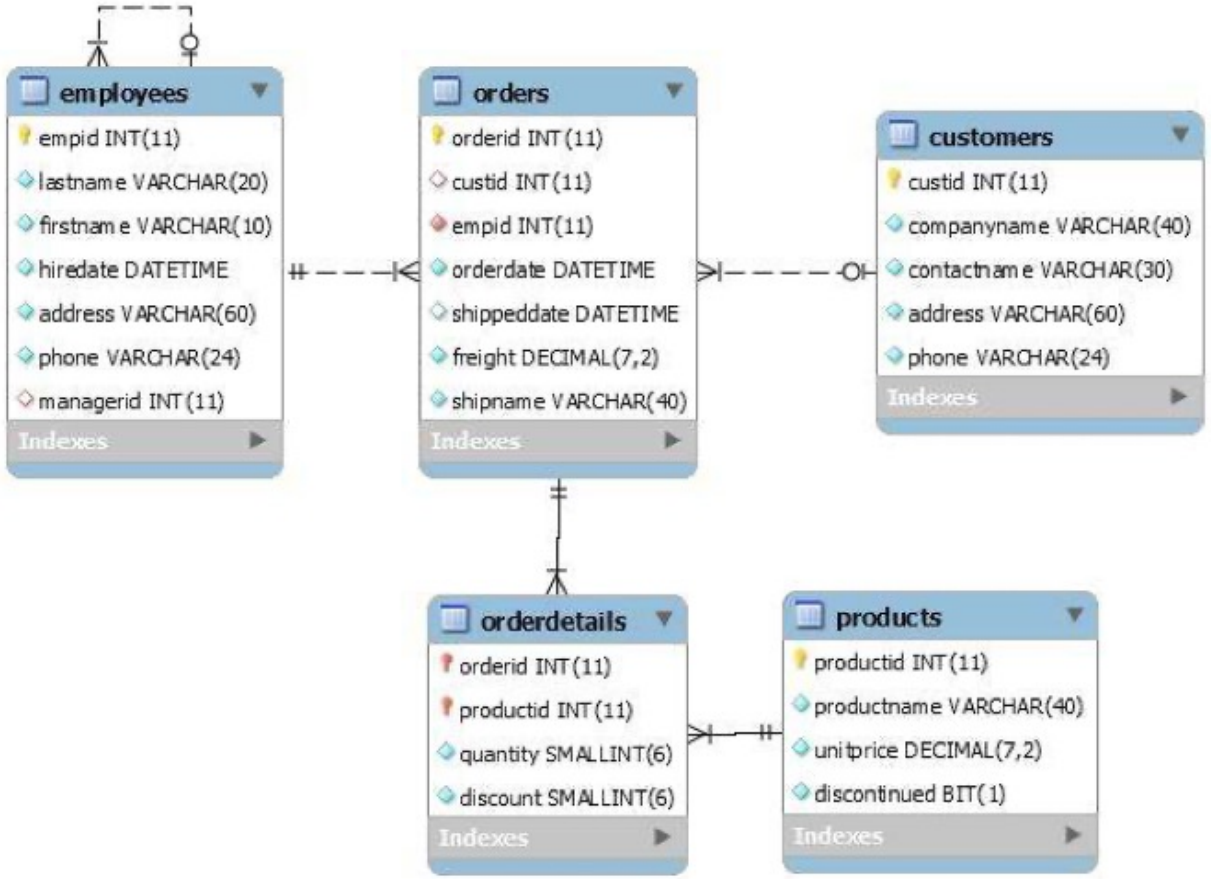
- Write a query that returns customers (company names) and the details of their orders (orderid and orderdate), including customers who placed no orders.

Case: Store System



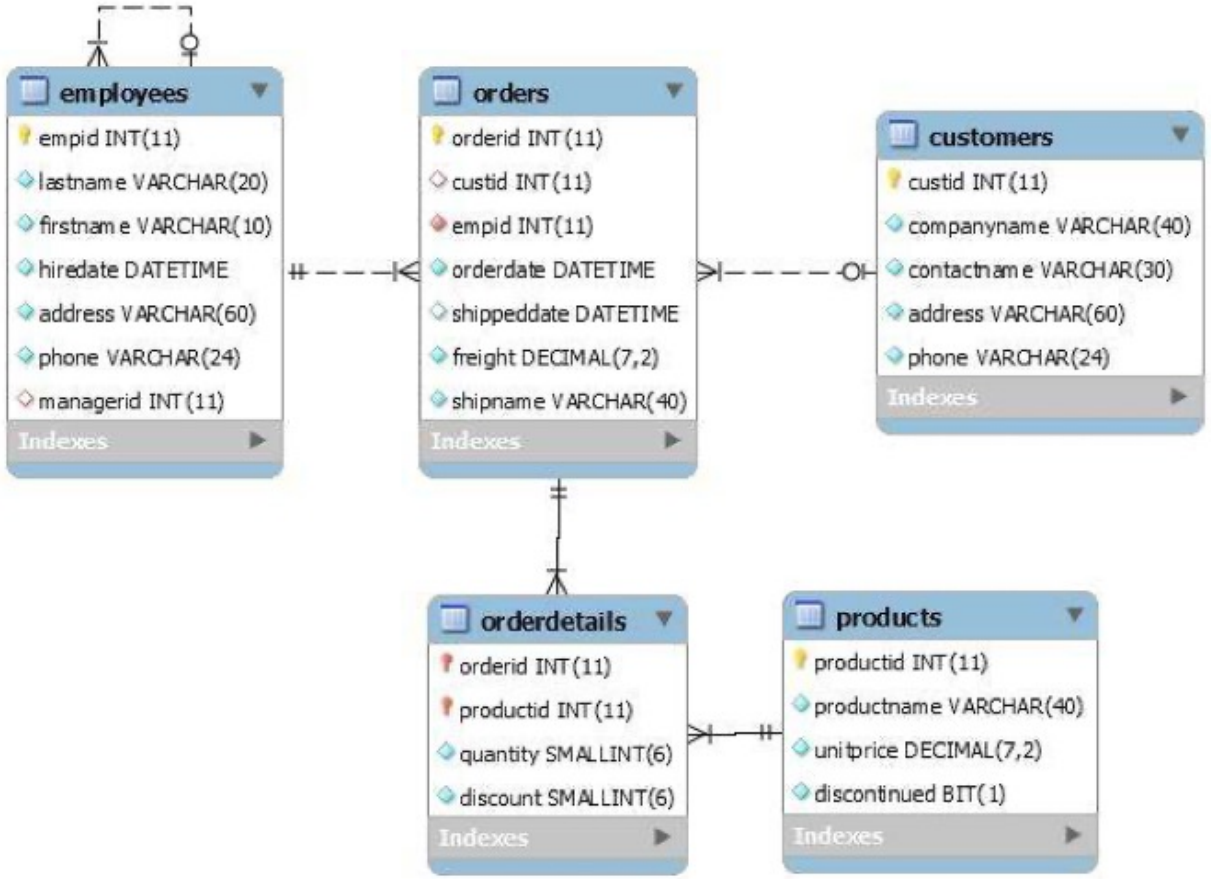
- Write a query that returns the first name and last name of employees whose manager was hired prior to 01/01/2002.

Case: Store System



- Write a query that returns customers (customer ID) whose company name is 'Google', and for each customer return the total number of orders and total quantities for all products that were not discontinued ('1' means discontinued, '0' not discontinued).

Case: Store System



- Write a query that returns the ID and company name of customers who placed orders in 2007 but not in 2008.

ANSWER- Case: Store System

Q.2A. Write a query that returns customers (company names) and the details of their orders (orderid and orderdate), including customers who placed no orders.

(3 marks)

```
SELECT C.companyname, O.orderid, O.orderdate
FROM Customers AS C LEFT OUTER JOIN Orders AS O
ON O.custid = C.custid;
```

Q.2B. Write a query that returns the first name and last name of employees whose manager was hired prior to 01/01/2002.

(4 marks)

```
SELECT E.firstname, E.lastname
FROM Employees AS E INNER JOIN Employees AS MNGR
ON E.managerid = MNGR.empid
WHERE MNGR.hiredate < '20020101';
```

Q.2C. Write a query that returns customers (customer ID) whose company name is 'Google', and for each customer return the total number of orders and total quantities for all products that were not discontinued ('1' means discontinued, '0' not discontinued).

(5 marks)


```
SELECT C.custid, COUNT(O.orderid) AS numorders, SUM(OD.quantity) AS
totalqty
FROM Customers AS C
  JOIN Orders AS O
    ON O.custid = C.custid
  JOIN OrderDetails AS OD
    ON OD.orderid = O.orderid
  JOIN Products AS P
    ON OD.productid = P.productid
WHERE C.company = 'Google'
  AND P.discontinued = '0'
GROUP BY C.custid;
```

ANSWER – Case: Store System

Q.2D. Write a query that returns the ID and company name of customers who placed orders in 2007 but not in 2008.

(8 marks)

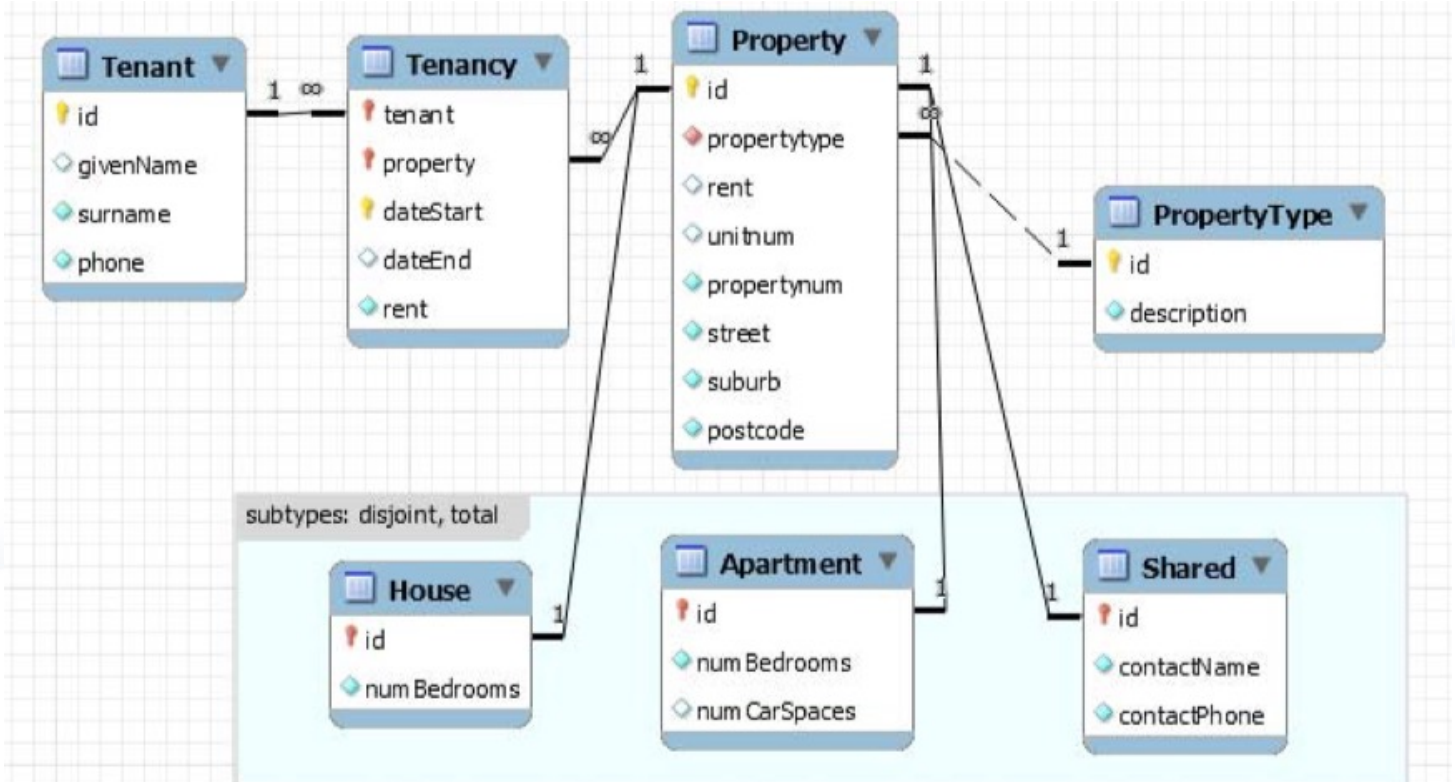
```
SELECT custid, companyname
FROM Customers
WHERE custid IN
  (SELECT custid
   FROM Orders
   WHERE orderdate >= '20070101'
     AND orderdate < '20080101')
  AND custid NOT IN
  (SELECT custid
   FROM Orders
   WHERE orderdate >= '20080101'
     AND orderdate < '20090101');

--    ALSO / OR

SELECT custid, companyname
FROM Customers INNER JOIN ORDERS O1
ON Customers.custid = O1.custID
WHERE YEAR(orderdate) = 20007;
WHERE NOT EXITS
    (SELECT *
     FROM Customers INNER JOIN ORDERS O2
     ON Customers.custid = O2.custID
     WHERE YEAR(orderdate) = 2018
     AND O1.custid = O2.custID);
```

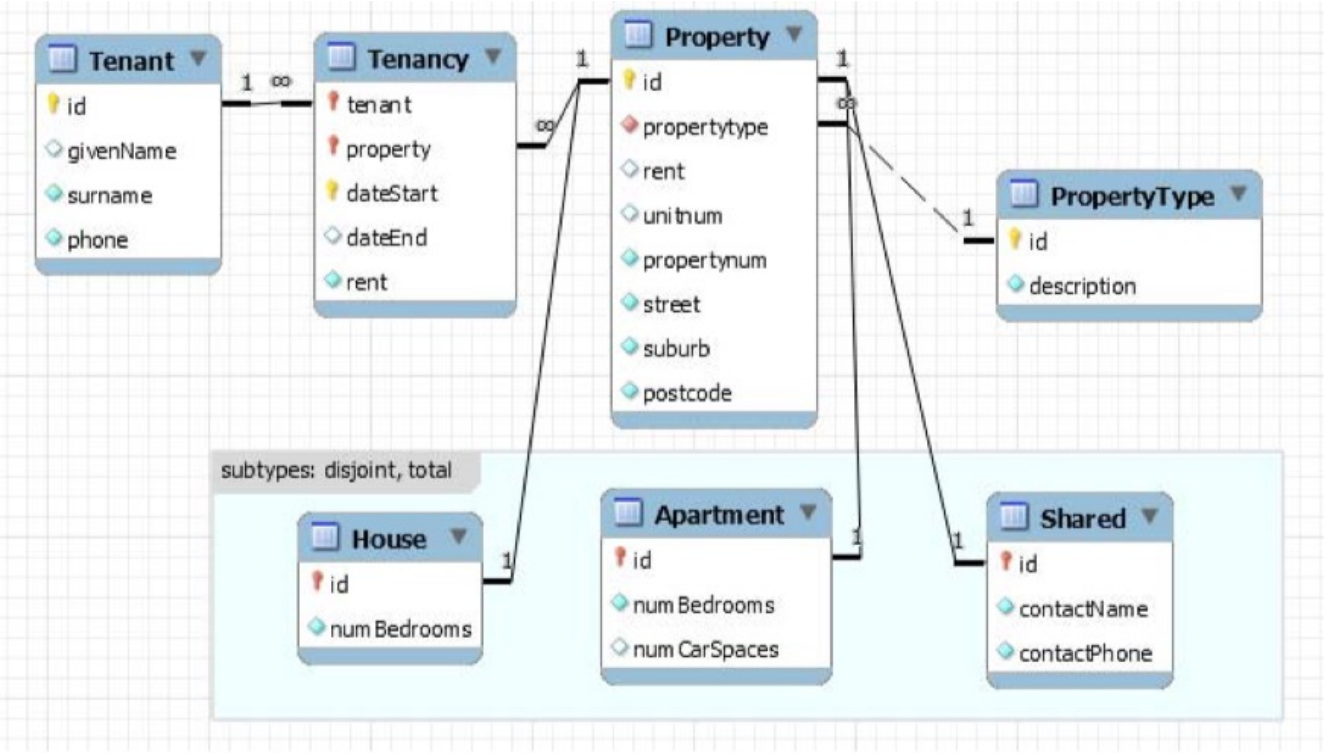
Case: Student Real-estate System

- Consider the following data model and sample data (not all data is shown) for a student real-estate system.
- At any given time, an individual property is empty or occupied by one tenancy. Note that the rent charged for a particular tenancy, especially one in the past, may be different to the current advertised rent on that property.



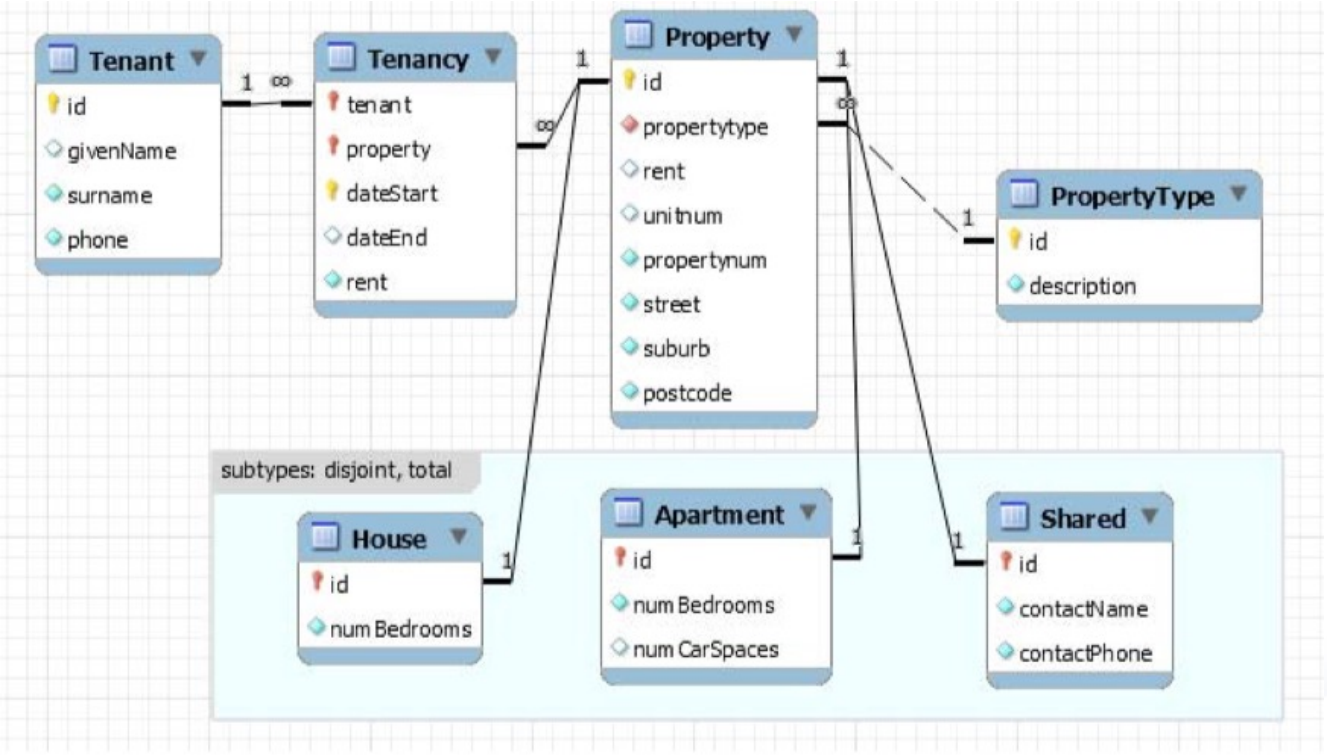
HD·EDUCATION

Case: Student Real-estate System



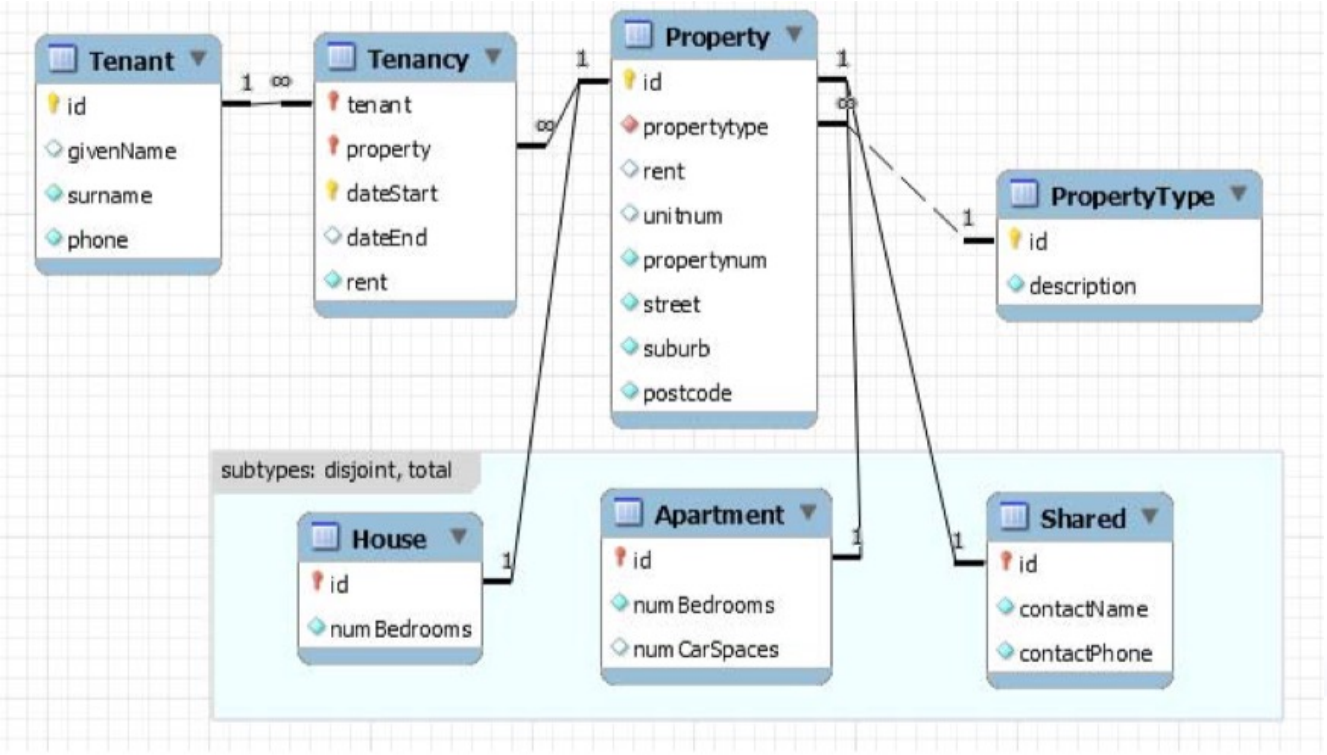
- A) Which tenants have paid us the most rent?
- List the top 5 payers, and how much rent each has paid in total.
- B) What is the longest given name among our tenants?
- If several names tie for first place, list them all.
- C) List the addresses of the houses and apartments which only have one bedroom.
- D) List the names of tenants who have rented at least two different types of properties.

Case: Student Real-estate System



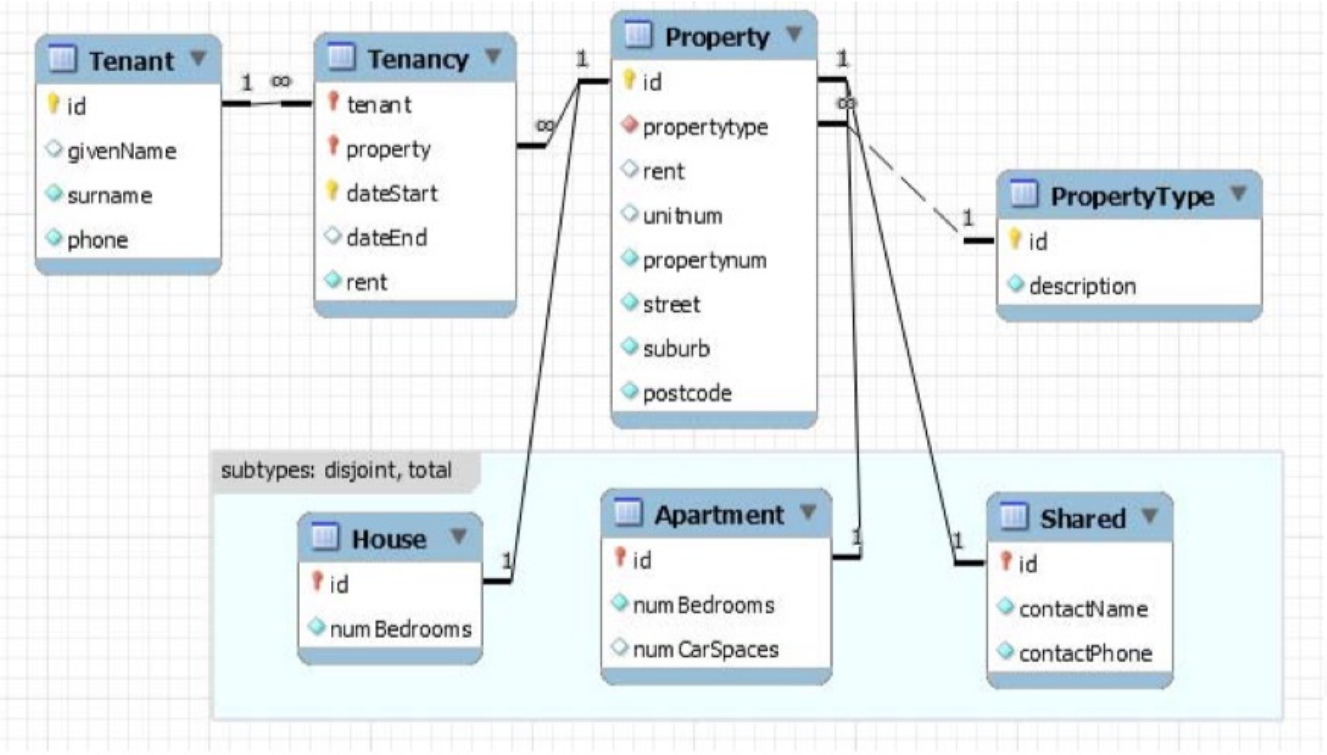
- A) Which tenants have paid us the most rent?
- List the top 5 payers, and how much rent each has paid in total.

Case: Student Real-estate System



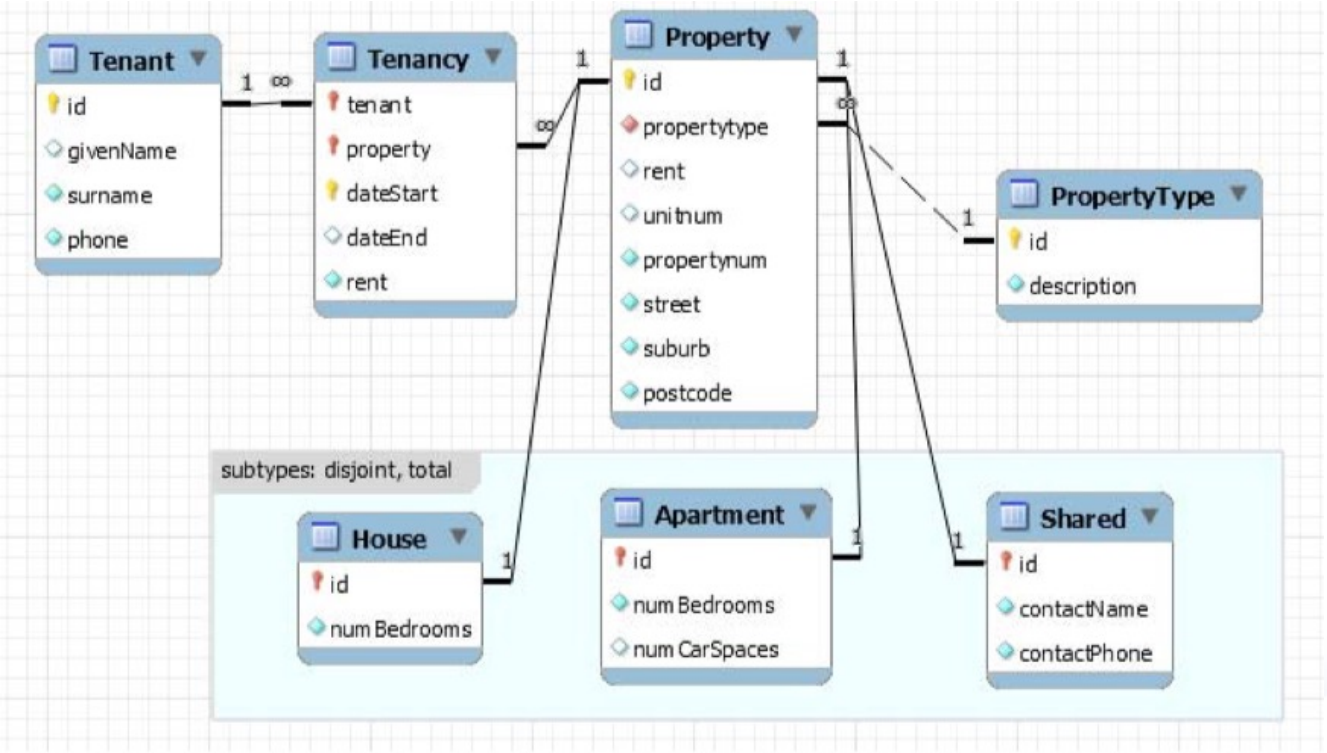
- B) What is the longest given name among our tenants?
- If several names tie for first place, list them all.

Case: Student Real-estate System



- C) List the addresses of the houses and apartments which only have one bedroom.

Case: Student Real-estate System



- D) List the names of tenants who have rented at least two different types of properties.

ANSWER-Case: Student Real-estate System

```sql
SELECT
    CONCAT(givenName, ' ', surname), SUM(rent) AS totalPayment
FROM
    Tenant
        INNER JOIN
    Tenancy ON Tenant.id = Tenancy.tenant
WHERE
    dateEnd <= CURDATE()
GROUP BY givenName , surname
ORDER BY SUM(rent) DESC
LIMIT 5
```

```sql
SELECT
    givenName
FROM
    Tenant
WHERE
    LENGTH(givenName) = (
        SELECT
            MAX(LENGTH(givenName))
        FROM
            Tenant
    )
```

ANSWER-Case: Student Real-estate System

```
SELECT
    CONCAT(
        IF(ISNULL(unitnum), '', CONCAT(unitnum, ' ')),
        street,
        ' ',
        suburb
    )
FROM
    Property
        LEFT JOIN
    Apartment ON Property.id = Apartment.id
        LEFT JOIN
    House ON Property.id = House.id
WHERE
    Apartment.numberBed = 1
    OR House.numberBed = 1
```

```
SELECT
    CONCAT(givenName, ' ', surname)
FROM
    Tenant
        INNER JOIN
    Tenancy ON Tenant.id = Tenancy.tenant
        INNER JOIN
    Property ON Tenancy.property = Property.id
GROUP BY Tenant.id
HAVING COUNT(DISTINCT propertyType) >= 2
```