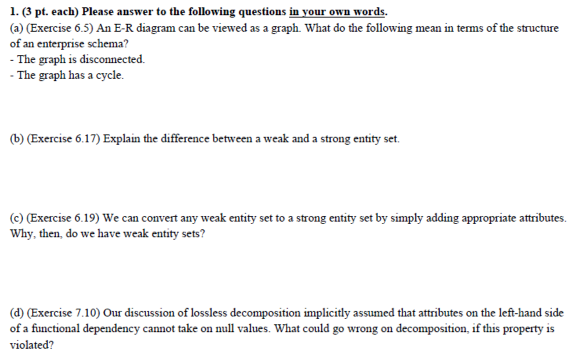
Homework Assignment 3

Due: 11:59PM, April 24, 2025

1.

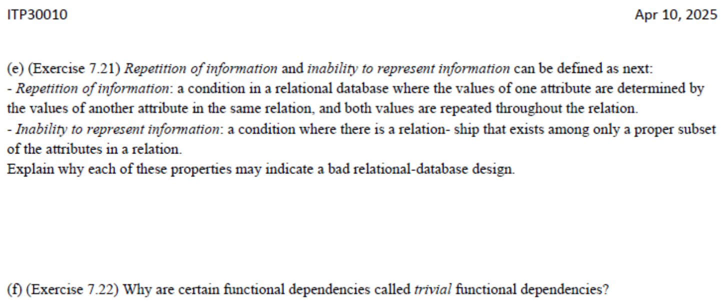


1. (3 pt. each) Please answer to the following questions in your own words.
(a) (Exercise 6.5) An E-R diagram can be viewed as a graph. What do the following mean in terms of the structure of an enterprise schema?
- The graph is disconnected.
- The graph has a cycle.

(b) (Exercise 6.17) Explain the difference between a weak and a strong entity set.

(c) (Exercise 6.19) We can convert any weak entity set to a strong entity set by simply adding appropriate attributes. Why, then, do we have weak entity sets?

(d) (Exercise 7.10) Our discussion of lossless decomposition implicitly assumed that attributes on the left-hand side of a functional dependency cannot take on null values. What could go wrong on decomposition, if this property is violated?



ITP30010 Apr 10, 2025

(e) (Exercise 7.21) *Repetition of information* and *inability to represent information* can be defined as next:
- *Repetition of information*: a condition in a relational database where the values of one attribute are determined by the values of another attribute in the same relation, and both values are repeated throughout the relation.
- *Inability to represent information*: a condition where there is a relation- ship that exists among only a proper subset of the attributes in a relation.
Explain why each of these properties may indicate a bad relational-database design.

(f) (Exercise 7.22) Why are certain functional dependencies called *trivial* functional dependencies?

Answer 1.

a. **Disconnected,** There is no connection between entities, thus which mean part of schema is isolated with other part.
Cycle, A path of relationships exists among entity sets where the path starts and ends same entity.

b. **Strong Entity,** An entity set that has a PK(Primary Key) and can be uniquely identified by its own attributes without depend on other entity.
Weak Entity, An entity set that dose not have sufficient primary key. It must rely on strong entity through a foreign key and partial key to be uniquely identified.

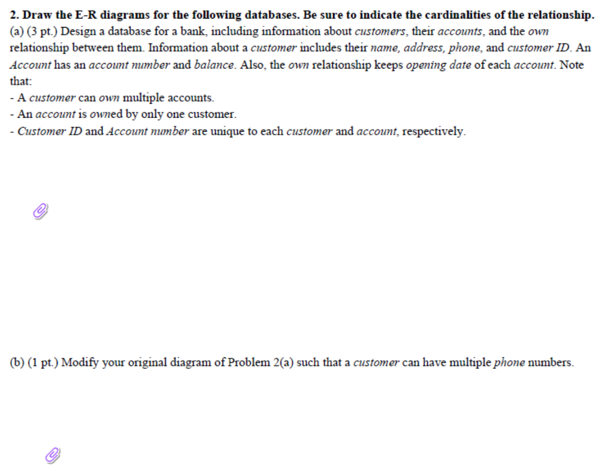
c. In real world modeling, some entities (like order list or a room in a building) do not make sense without their related entity. Therefore, weak entity sets are useful because explicitly represent existential dependency on another entity.

d. Null values are not considered equal to any value which means Null values cannot ensure that functional dependencies hold, thus decomposed relations may not preserve the original data when joined leading to lossy decomposition.

e. **Repetition,** repeated value leads to ‘redundancy’, this causes (update, insertion, and deletion) anomalies.
Inability, if schema groups together unrelated attributes, it may prevent us from representing valid real-world scenarios.

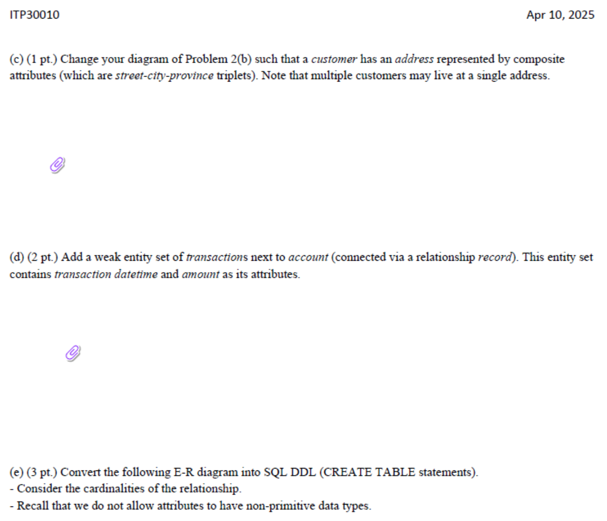
f. A functional dependency X → Y is called **trivial** if Y is a subset of X. In this case, the dependency always holds in any relation instance and thus does not impose any meaningful constraint on the data. Therefore, it is called a **trivial functional dependency**.

2.



2. Draw the E-R diagrams for the following databases. Be sure to indicate the cardinalities of the relationship.
(a) (3 pt.) Design a database for a bank, including information about *customers*, their *accounts*, and the *own* relationship between them. Information about a *customer* includes their *name, address, phone*, and *customer ID*. An *Account* has an *account number* and *balance*. Also, the *own* relationship keeps *opening date* of each *account*. Note that:
- A *customer* can *own* multiple accounts.
- An *account* is *own*ed by only one customer.
- *Customer ID* and *Account number* are unique to each *customer* and *account*, respectively.

(b) (1 pt.) Modify your original diagram of Problem 2(a) such that a *customer* can have multiple *phone* numbers.



ITP30010 Apr 10, 2025

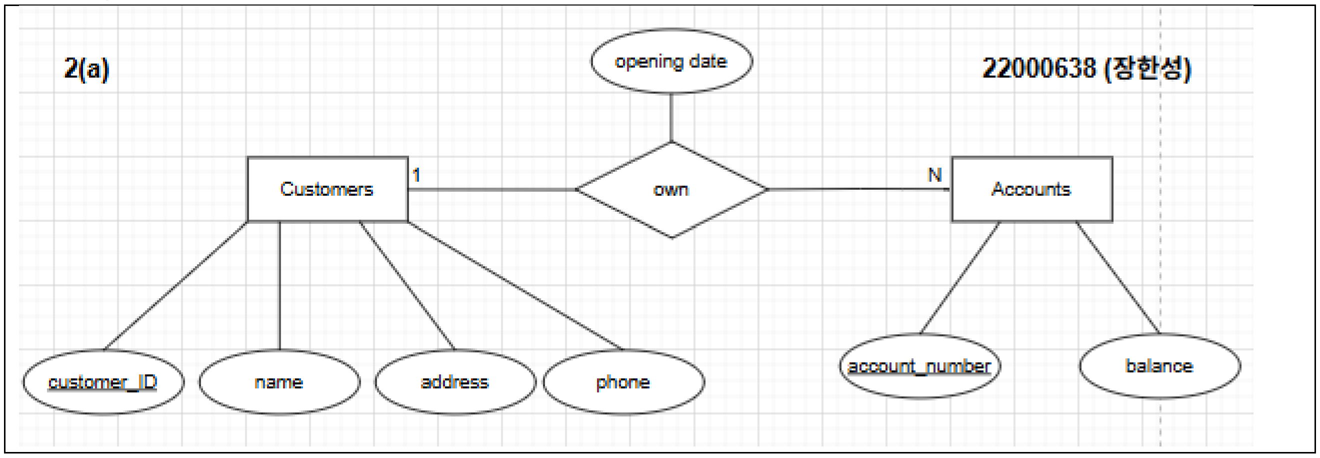
(c) (1 pt.) Change your diagram of Problem 2(b) such that a *customer* has an *address* represented by composite attributes (which are *street-city-province* triplets). Note that multiple customers may live at a single address.

(d) (2 pt.) Add a weak entity set of *transaction*s next to *account* (connected via a relationship *record*). This entity set contains *transaction datetime* and *amount* as its attributes.

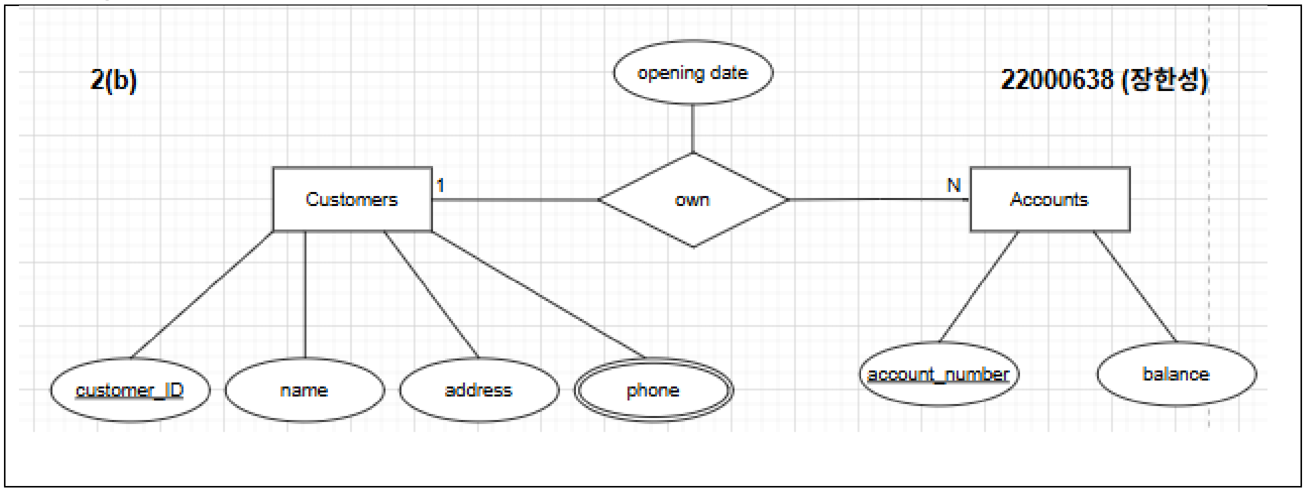
(e) (3 pt.) Convert the following E-R diagram into SQL DDL (CREATE TABLE statements).
- Consider the cardinalities of the relationship.
- Recall that we do not allow attributes to have non-primitive data types.

Answer 2

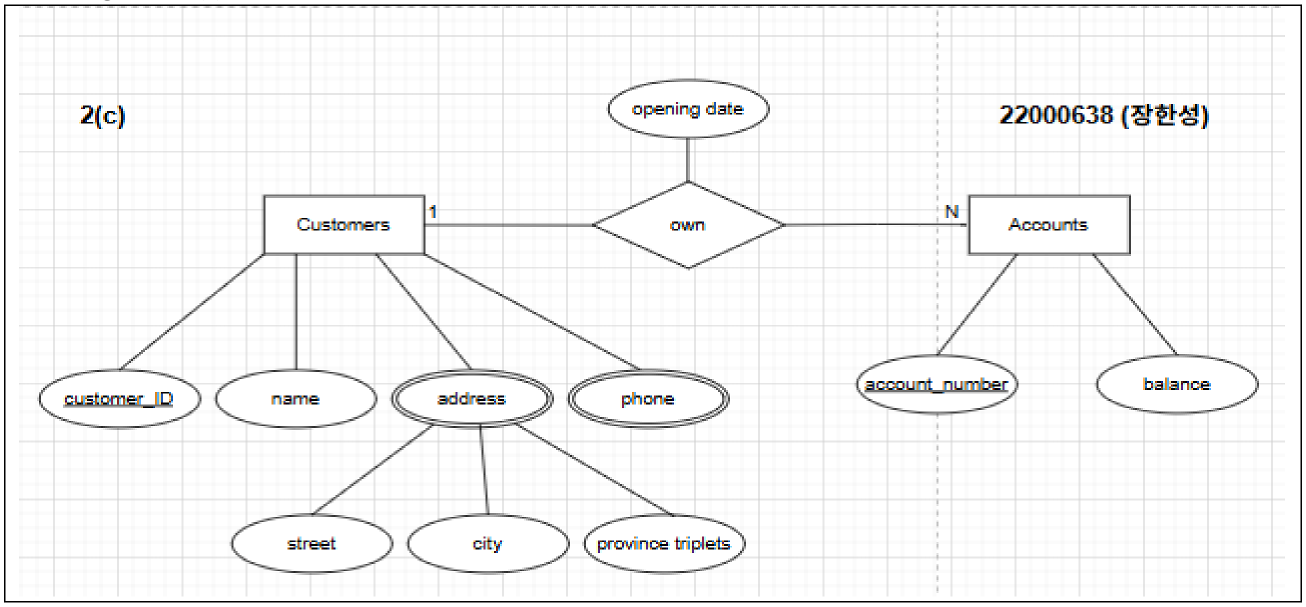
[E-R diagram / 2(a)]



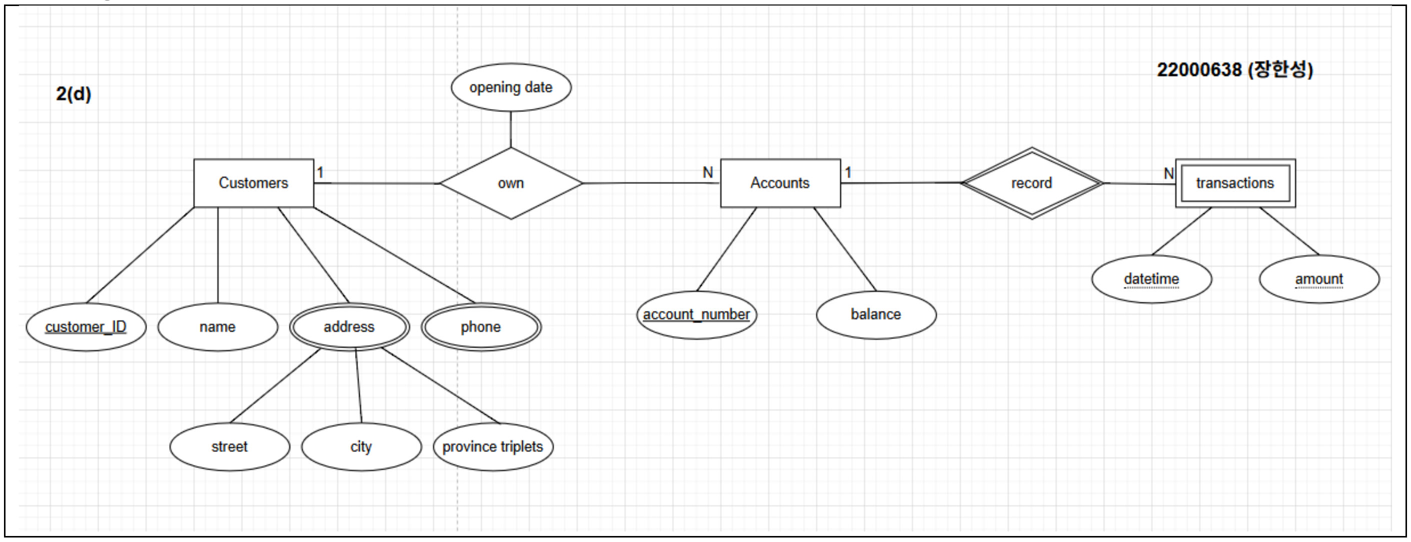
[E-R diagram / 2(b)]



[E-R diagram / 2(c)]



[E-R diagram / 2(d)]



[E-R diagram / 2(e)]

[1. Address]

```
CREATE TABLE Address (
  address_id INT PRIMARY KEY,
  street VARCHAR(50),
  city VARCHAR(50),
  province_triplets VARCHAR(50)
);
```

[2. Customers]

```
CREATE TABLE Customers (
  customer_ID INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  address_id INT,
  FOREIGN KEY (address_id) REFERENCES Address(address_id)
);
```

[3. CustomerPhones (다치 속성 분리)]

```
CREATE TABLE CustomerPhones (
  customer_ID INT,
  phone VARCHAR(50),
  PRIMARY KEY (customer_ID, phone),
  FOREIGN KEY (customer_ID) REFERENCES Customers(customer_ID)
);
```

[4. Accounts]

```
CREATE TABLE Accounts (
  account_number INT PRIMARY KEY,
  balance DOUBLE,
  customer_ID INT,
  FOREIGN KEY (customer_ID) REFERENCES Customers(customer_ID)
);
```

[5. Transactions (약한 엔터티)]

```
CREATE TABLE Transactions (
  account_number INT,
  datetime DATETIME,
  amount INT,
  PRIMARY KEY (account_number, datetime, amount),
  FOREIGN KEY (account_number) REFERENCES Accounts(account_number) ON
DELETE CASCADE
);
```

3. Normalization(정규화)

(a) Is every relation in 3NF also in BCNF? If yes, explain why. If no, given a counter example.

Answer:

No, not every relation in 3NF is also in BCNF.

A relation can be in 3NF even if there is a functional dependency where the determinant is not a candidate key, as long as the dependent attribute is a prime attribute or the dependency is trivial. However, BCNF requires that every non-trivial functional dependency must have a candidate key as its determinant.

Counter Example:
Consider a relation R(student_id, course_id, instructor)

- Assume (student_id) is the only candidate key
- Suppose there is a functional dependency: course_id → instructor
- This satisfies 3NF because instructor is not transitively dependent on the whole key
- But 'course_id' is not a candidate key, so this violates BCNF

(b) Is every relation in 4NF also in BCNF? If yes, explain why. If no, given a counter example.

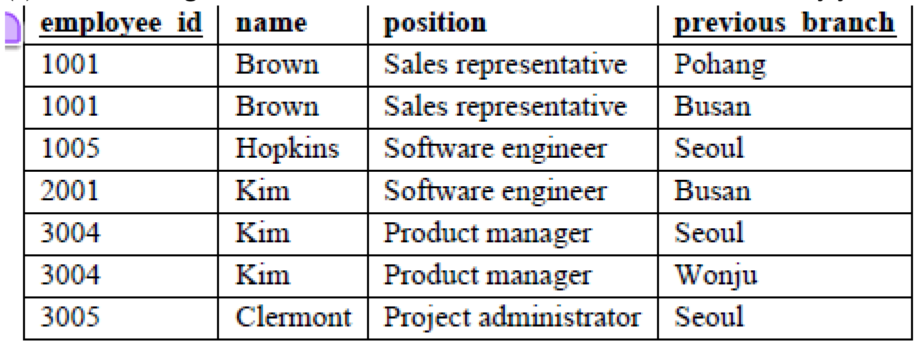
Answer:

Yes, every relation in 4NF is also in BCNF.

4NF requires that all non-trivial multiple dependencies (MVDs) be dependent only on candidate keys, It is a stronger regular form than BCNF, which requires candidate key conditions for all functional dependencies (FD).

Therefore, 4NF-satisfying relationships always satisfy BCNF as well.

(c) The following relation violates {1NF, 2NF, 3NF, 4NF, BCNF}? Justify your answer.



employee_id	name	position	previous_branch
1001	Brown	Sales representative	Pohang
1001	Brown	Sales representative	Busan
1005	Hopkins	Software engineer	Seoul
2001	Kim	Software engineer	Busan
3004	Kim	Product manager	Seoul
3004	Kim	Product manager	Wonju
3005	Clermont	Project administrator	Seoul

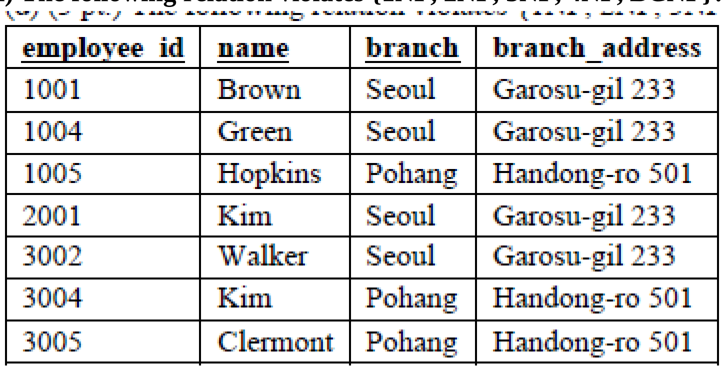
Answer:

4NF violation.

The following table violates 4NF due to the presence of multivalued dependencies:
Each employee can be associated with multiple previous branches (e.g., Busan, Pohang) and also have a specific position, but the position and previous_branch are independent, causing repetition (cartesian product).

[Example]
employee_id = 1001 -> "Sales representative" * {Pohang, Busan}

(d) The following relation violates {1NF, 2NF, 3NF, 4NF, BCNF}? Justify your answer.



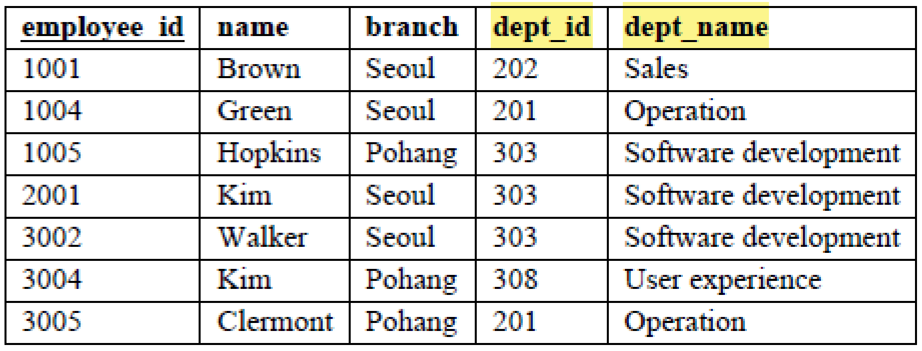
employee_id	name	branch	branch_address
1001	Brown	Seoul	Garosu-gil 233
1004	Green	Seoul	Garosu-gil 233
1005	Hopkins	Pohang	Handong-ro 501
2001	Kim	Seoul	Garosu-gil 233
3002	Walker	Seoul	Garosu-gil 233
3004	Kim	Pohang	Handong-ro 501
3005	Clermont	Pohang	Handong-ro 501

Answer:

2NF violation.

This relationship is based on the assumption that employee_id is the default key,
branch_address is not the entire default key (employee_id), Depends only on the property branch.
This corresponds to partial dependence, It violates the second normal type (2NF).

(e) The following relation violates {1NF, 2NF, 3NF, 4NF, BCNF}? Justify your answer.



employee_id	name	branch	dept_id	dept_name
1001	Brown	Seoul	202	Sales
1004	Green	Seoul	201	Operation
1005	Hopkins	Pohang	303	Software development
2001	Kim	Seoul	303	Software development
3002	Walker	Seoul	303	Software development
3004	Kim	Pohang	308	User experience
3005	Clermont	Pohang	201	Operation

Answer:

3NF violation.

there is a transitive dependency:

- employee_id → dept_id
- dept_id → dept_name

  ⇒ so, employee_id → dept_name (indirect dependency)

Since dept_name is a non-prime attribute that is transitively dependent on the primary key, this violates Third Normal Form (3NF).

4. SQL Queries

Launch and access the MySQL databases distributed with the class virtual machine. Below uses the "***sakila***" database (DVD rental database), which consists of 16 tables regarding movie inventory, actors, customers, rental history, payment information, *etc*. For each of the following questions, find the answer based on the information recorded in the database and write a query that shows how you obtained the answer.

Question 4(a), How many *stores* are found in the database?
Answer 4(a)

[Answer question]	**[SQL query answer]**
2	SELECT *Count*(*) FROM sakila.store;

Question 4(b) How many unique *last names* are found in the *actor* relation?
Answer 4(b):

[Answer question]	**[SQL query answer]**

121	SELECT *COUNT*(DISTINCT last_name)
	FROM actor;

Question 4(c) According to the database, how many *inventories* (DVDs) have not been returned (inventories that have not been returned do not have *return_date*)?
Answer 4(c):

[Answer question]	**[SQL query answer]**

183	SELECT *Count*(inventory.inventory_id)
	FROM inventory
	JOIN rental ON inventory.inventory_id = rental.inventory_id
	WHERE rental.return_date IS NULL;

Question 4(d) How many distinct *customers* have rented a movie title(s) from *staff_id*=1?
Answer 4(d):

[Answer question]	**[SQL query answer]**

8,057	SELECT *COUNT*(customer_id) AS num_rent
	FROM payment
	WHERE payment.staff_id = 1
	GROUP BY customer_id;

Question 4(e) How many distinct films rated 'PG' are available?
Answer 4(e):

[Answer question]	**[SQL query answer]**

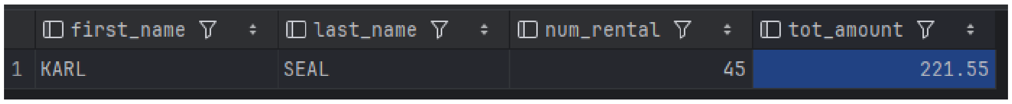
183	SELECT *COUNT*(DISTINCT f_pg.film_id) AS Avilable_PG
	FROM (SELECT film_id, rating
	FROM film
	WHERE rating = 'PG') AS f_pg, rental AS r, inventory as i
	WHERE (r.return_date IS NOT NULL) AND f_pg.film_id = i.film_id
	AND i.inventory_id = r.inventory_id;

Question 4(f) How many active customers are living in the district of England?
Answer 4(f):

[Answer question]	**[SQL query answer]**

6	SELECT *COUNT*(*)
	FROM customer AS c, address AS a
	WHERE (c.active = 1 AND c.address_id = a.address_id AND a.district LIKE '%England%');

Question 4(g) Considering the rental history (*rental*) and payment history (*payment*), who has paid the largest amount of money for renting movies? List the *first* and *last name* of the *customer*, the *total number* of movie rentals, and *total amount* of money s/he has paid.
Answer 4(g):



	first_name	last_name	num_rental	tot_amount
1	KARL	SEAL	45	221.55

[SQL query answer]

```
SELECT first_name, last_name, coast_list.num_rental, coast_list.tot_amount
FROM customer,
    (
        SELECT r.customer_id, COUNT(DISTINCT r.rental_id) AS num_rental,
p.tot_amount
        FROM rental AS r
                JOIN (SELECT customer_id, SUM(amount) AS tot_amount
                    FROM payment
                    GROUP BY customer_id) AS p ON p.customer_id = r.customer_id
#WHERE r.customer_id = p.customer_id
        GROUP BY customer_id     ) AS coast_list
```

```
WHERE coast_list.customer_id = customer.customer_id
ORDER BY tot_amount DESC LIMIT 1;
```

Question 4(h) List three most common *categories* of film available at *store_id*=2 (if a store has multiple copies of the same film, consider each copy as an individual inventory).
Answer 4(h):

[Answer question]	**[SQL query answer]**

1. **Sports**
2. **Documentary**
3. **Animation**
	```
	SELECT name
	FROM category AS c, (SELECT category_id, COUNT(category_id) AS num_category
	          FROM film_category AS fc,(SELECT film_id
	                    FROM inventory AS i, (
	                      SELECT inventory_id
	                      FROM rental
	                      WHERE return_date IS NOT NULL
	                    ) AS r
	                    WHERE (i.store_id = 2 AND i.inventory_id = r.inventory_id)) AS i
	          WHERE fc.film_id = i.film_id
	          GROUP BY  category_id
	          ORDER BY num_category DESC LIMIT 3
	) common_ca
	WHERE c.category_id = common_ca.category_id;
	```

Question 4(i) What is the *title* of the movie that has the longest description (*film_text.description*) among the rental store with *store_id*=2 has?
Answer 4(i):

[Answer question]	**[SQL query answer]**

CANDIDATE PERDITION	
	```
	SELECT DISTINCT ft.title
	FROM film_text AS ft
	    JOIN film_text AS f ON ft.film_id = f.film_id
	    JOIN inventory AS i ON f.film_id = i.film_id
	WHERE LENGTH(ft.description) >=
	   (SELECT MAX(LENGTH(description)) FROM film_text)
	 AND i.store_id = 2;
	```

Question 4(j) Which of the films starred by "FRED COSTNER" rented the most?
Answer 4(j):

[Answer question]	**[SQL query answer]**

BROTHERHOOD BLANKET	SELECT title, *COUNT*(title) AS Num_inven
	FROM rental AS ren, (SELECT inventory_id, ft.title
	FROM inventory AS inv, (SELECT film_id
	FROM film_actor AS fa, (SELECT actor_id
	FROM actor AS a
	WHERE a.first_name LIKE 'FRED' AND a.last_name LIKE 'COSTNER') AS a
	WHERE fa.actor_id = a.actor_id) AS f_id, film_text AS ft
	WHERE inv.film_id = f_id.film_id AND f_id.film_id = ft.film_id) AS inv_mv
	WHERE ren.inventory_id = inv_mv.inventory_id
	GROUP BY title
	ORDER BY Num_inven DESC LIMIT 1
	;

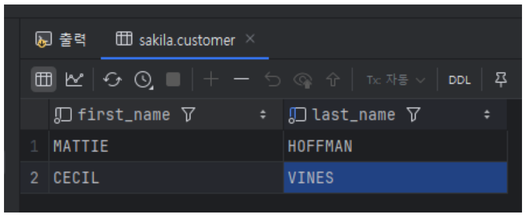
Question 4(k) Using the *'customer_list'* view, list all names of people whose address is in the city of 'London'.
Answer 4(k):

[Answer question]	**[SQL query answer]**

1. **MATTIE HOFFMAN**
2. **CECIL VINES**
	SELECT name
	FROM customer_list
	WHERE city LIKE '%London%';

Question 4(l) Write a query *that uses only tables (does not use any views)* and returns the same information as in the previous problem (Problem (k)).
Answer 4(l):

[Answer question]	**[SQL query answer]**

 first_name, last_name
1 MATTIE HOFFMAN
2 CECIL VINES
	SELECT first_name, last_name
	FROM customer AS cust, (SELECT address_id
	FROM address AS addr, (SELECT city_id
	FROM city
	WHERE city LIKE
	'London') AS city_check
	WHERE addr.city_id =
	city_check.city_id) AS addr
	WHERE cust.address_id = addr.address_id
	;