Answers 3.2.

2.

actor		
PK	actor_id	integer
	first_name	character varying(45)
	last_name	character varying(45)
	last_update	timestamp without time zone

category		
PK	category_id	integer
	name	character varying(25)
	last_update	timestamp without time zone

country		
PK	country_id	integer
	country	character varying(50)
	last_update	timestamp without time zone

film		
PK	film_id	integer
	title	character varying(255)
	description	text
	release_year	integer
FK	language_id	smallint
	rental_duration	smallint
	rental_rate	numeric
	length	smallint
	replacement_cost	numeric
	rating	USER-DEFINED
	last_update	timestamp without time zone
	special_features	ARRAY
	fulltext	tsvector

address		
PK	address_id	integer
	address	character varying(50)
	address2	character varying(50)
	district	character varying(20)
FK	city_id	smallint
	postal_code	character varying(10)
	phone	character varying(20)
	last_update	timestamp without time zone

city		
PK	city_id	integer
	city	character varying(50)
FK	country_id	smallint
	last_update	timestamp without time zone

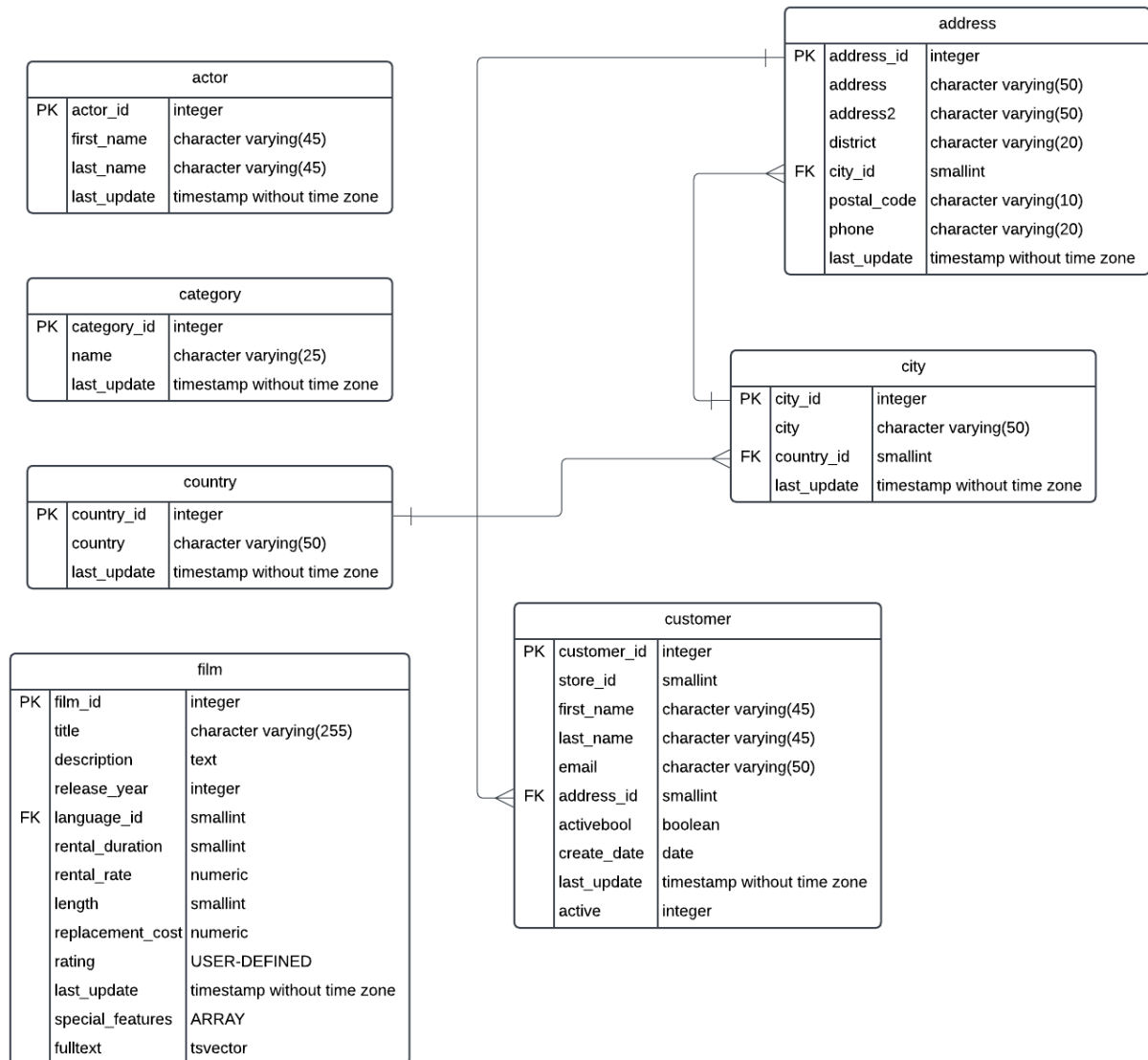| customer		
PK	customer_id	integer
	store_id	smallint
	first_name	character varying(45)
	last_name	character varying(45)
	email	character varying(50)
FK	address_id	smallint
	activebool	boolean
	create_date	date
	last_update	timestamp without time zone
	active	integer

```
CREATE TABLE `customer` (
 `customer_id` integer,
 `store_id` smallint,
 `first_name` character varying(45),
 `last_name` character varying(45),
 `email` character varying(50),
 `address_id` smallint,
 `activebool` boolean,
 `create_date` date,
 `last_update` timestamp without time zone,
 `active` integer,
 PRIMARY KEY (`customer_id`)
```

);

3: Create the First Draft of a Data Dictionary

Looking at your ERD, the Rockbuster database appears to follow a star schema:

- A central fact table (film) connects to dimension tables like actor, category, customer, city, and country.
- This schema simplifies query performance for analytics, as dimensions directly relate to the fact table without significant normalization (as in a snowflake schema).

3b.

Fact Tables

1. Film
This is the main fact table that stores information about films.

Column Name	Data Type	Description
film_id	integer	Unique identifier for each film.
title	character varying(255)	Title of the film.
description	text	A short summary or synopsis of the film.
release_year	integer	Year the film was released.
language_id	smallint	References the `language` table to specify the language of the film.
rental_duration	smallint	Number of days the film can be rented for.
rental_rate	numeric	Cost to rent the film for the rental duration.
replacement_cost	numeric	Cost to replace the film if lost or damaged.
rating	USER-DEFINED	Rating of the film (e.g., G, PG, R).
special_features	ARRAY	Special features included with the film (e.g., deleted scenes, commentaries).
fulltext	tsvector	Full-text search vector for the film description.

Dimension Tables

1. Actor
Stores information about the actors in the films.

Column Name	Data Type	Description
actor_id	integer	Unique identifier for each actor.
first_name	character varying(45)	First name of the actor.

last_name	character varying(45)	Last name of the actor.
last_updat e	timestamp without time zone	Date and time when the actor's details were last updated.

2. Category
Stores information about the categories of films.

Column Name	Data Type	Description
category_i d	integer	Unique identifier for each category.
name	character varying(25)	Name of the category.
last_updat e	timestamp without time zone	Date and time when the category details were last updated.

3. Customer
Stores information about customers who rent films.

Column Name	Data Type	Description
customer_i d	integer	Unique identifier for each customer.
store_id	smallint	ID of the store the customer is associated with.
first_name	character varying(45)	First name of the customer.
last_name	character varying(45)	Last name of the customer.
email	character varying(50)	Email address of the customer.
address_id	smallint	References the `address` table to specify the customer's address.
activebool	boolean	Indicates whether the customer is active.
create_date	date	Date the customer record was created.
last_update	timestamp without time zone	Date and time when the customer details were last updated.
active	integer	Specifies whether the customer is active (1 = active, 0 = inactive).

4. Address
Stores address information for customers.

Column Name	Data Type	Description
address_id	integer	Unique identifier for each address.
address	character varying(50)	First line of the address.
address2	character varying(50)	Second line of the address (optional).
district	character varying(20)	District or region of the address.
city_id	smallint	References the `city` table to specify the city of the address.
postal_cod e	character varying(10)	Postal code for the address.

phone	character varying(20)	Phone number associated with the address.
last_update	timestamp without time zone	Date and time when the address details were last updated.

5. City

Stores information about cities.

Column Name	Data Type	Description
city_id	integer	Unique identifier for each city.
city	character varying(50)	Name of the city.
country_id	smallint	References the `country` table to specify the country of the city.
last_updat e	timestamp without time zone	Date and time when the city details were last updated.

6. Country

Stores information about countries.

Column Name	Data Type	Description
country_id	integer	Unique identifier for each country.
country	character varying(50)	Name of the country.
last_updat e	timestamp without time zone	Date and time when the country details were last updated.

Step 4.

a.We need to calculate the total revenue for each actor based on the rental rate of the movies they acted in, and this requires linking actors to movies and rentals.

SQL joins are best to use to combine the data across these tables, ensuring an accurate and comprehensive revenue calculation.

Query:

SELECT

  a.first_name,

  a.last_name,

  SUM(f.rental_rate) AS total_revenue

FROM

  actor a

JOIN

  film_actor fa ON a.actor_id = fa.actor_id

JOIN

film f ON fa.film_id = f.film_id

JOIN

    inventory i ON f.film_id = i.film_id

JOIN

    rental r ON i.inventory_id = r.inventory_id

GROUP BY

    a.actor_id, a.first_name, a.last_name

ORDER BY

    total_revenue DESC

LIMIT 10;

Relavant Tables are :

- actor: gives the actor's name and ID (actor_id).
- film_actor: links actors to films using actor_id and film_id.
- film: contains details about each film, including rental_rate, which determines the revenue per rental.
- inventory: links each film to a specific inventory item (e.g., a physical copy of the movie).
- rental: tracks which inventory items were rented and when, helping us calculate how often a film was rented.

Here below you see the top 10 actors who generated the most revenue for Rockbuster, along with the total revenue they contributed. The total revenue was calculated based on the rental rates of the movies these actors starred in and the frequency of their rentals.

Rank	First Name	Last Name	Total Revenue (€)
1	Gina	Degeneres	2,469.47
2	Henry	Berry	1,983.88
3	Matthew	Carrey	1,935.22
4	Scarlett	Damon	1,848.28
5	Walter	Torn	1,829.60
6	Christian	Akroyd	1,800.48
7	Angela	Witherspoon	1,799.46
8	Helen	Voight	1,789.43
9	Mary	Keitel	1,767.26
10	Cameron	Zellweger	1,760.40

b. The majority of movies in the Rockbuster collection are in <mark>English, with a total of 1,000 movies</mark>.

The relevant tables are <mark>film,</mark> which contains details about each film, including <mark>language_id</mark>, and <mark>language,</mark> that maps the language_id to the language name. We need to JOIN the film table to the language table using the language_id.

Query :

```
SELECT

    l.name AS language,

    COUNT(f.film_id) AS movie_count

FROM

    film f

JOIN

    language l ON f.language_id = l.language_id

GROUP BY

    l.name

ORDER BY

    movie_count DESC

LIMIT 1;
```