Sql Question(SQL200_Task1)

Description: SQL200 – WorkSheet (Questions on Sakila database)

Q1 — Film prices From film, show: film_id, title, rental_rate Get only the films where rental_rate is 9.99 or 4.99. _____

Q2 — Film length + rating From film, show: title, length, rating Find films that are 90 to 120 minutes (inclusive) and rating is PG or PG-13. _____

Q3 — Actor last names From actor, show: actor_id, first_name, last_name Find actors whose last_name starts with S OR ends with N. _____

Q4 — Active customers + email filter From customer, show: customer_id, first_name, last_name, email Find active customers whose email contains ".org" OR ".net". _____

Q5 — Inactive customers in store 1 From customer, show: customer_id, store_id, active Find customers from store 1 who are not active. _____

Q6 — Payment amount + date range From payment, show: payment_id, customer_id, amount, payment_date Find payments with amount between 2.00 and 5.00 and made in February 2007. _____

Q7 — Rentals not returned From rental, show: rental_id, rental_date, return_date, customer_id Find rentals where return_date is NULL. _____

Q8 — Address district + postal code present From address, show: address_id, address, district, postal_code Find addresses where district is Texas or California AND postal_code is not NULL. _____

Q9 — Replacement cost + exclude titles From film, show: film_id, title, replacement_cost Find films where replacement_cost is 12.99, 16.99, or 28.99 AND the title does NOT contain the letter A. _____

Q10 — Inventory logic challenge From inventory, show: inventory_id, film_id, store_id Find inventory items where: • (store_id = 1 AND film_id between 1 and 50) OR • (store_id = 2 AND film_id between 51 and 100)

Q1 — Film prices From film, show: film_id, title, rental_rate Get only the films where rental_rate is 9.99 or 4.99. _____

`film` table create karo

```
CREATE TABLE film (
  film_id SERIAL PRIMARY KEY,
  title VARCHAR(100),
  rental_rate NUMERIC(4,2)
);
```

data insert karo

```
INSERT INTO film (title, rental_rate) VALUES
('Avengers', 9.99),
('Inception', 4.99),
('Titanic', 2.99),
('Matrix', 9.99);
```

```
SELECT film_id, title, rental_rate
FROM film
WHERE rental_rate IN (9.99, 4.99);
```

Output

- 9.99
- 4.99

Q2 — Film length + rating From film, show: title, length, rating Find films that are 90 to 120 minutes (inclusive) and rating is PG or PG-13. _____

CREATE karo

```
CREATE TABLE film (
  film_id SERIAL PRIMARY KEY,
  title VARCHAR(100),
  rental_rate NUMERIC(4,2),
  length INT,
  rating VARCHAR(10)
);
```

Values (data) INSERT karo

```
INSERT INTO film (title, rental_rate, length, rating) VALUES
('Avengers', 9.99, 95, 'PG'),
('Inception', 4.99, 120, 'PG-13'),
('Titanic', 2.99, 130, 'R'),
('Matrix', 9.99, 100, 'PG'),
('Jumanji', 3.99, 90, 'PG'),
('Batman', 5.99, 110, 'PG-13');
```

Check data

```
SELECT * FROM film;

SELECT title, length, rating
FROM film
WHERE length BETWEEN 90 AND 120
AND rating IN ('PG', 'PG-13');
```

Q3 — Actor last names From actor, show: actor_id, first_name, last_name Find actors whose last_name starts with S OR ends with N. _____

CREATE karo

```
CREATE TABLE actor (
  actor_id SERIAL PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50)
);
```

INSERT karo

```
INSERT INTO actor (first_name, last_name) VALUES
('Amit', 'Sharma'),
('Ravi', 'Singh'),
('Neha', 'Khan'),
('Rahul', 'Saxena'),
('Pooja', 'Meen'),
('Karan', 'Verma'),
('Ankit', 'Saran');
```

Check data

```
SELECT * FROM actor;
```

query

```
SELECT actor_id, first_name, last_name
FROM actor
WHERE last_name LIKE 'S%'
  OR last_name LIKE '%N';
```

Q3 — Actor last names From actor, show: actor_id, first_name, last_name Find actors whose last_name starts with S OR ends with N. _____

CREATE karo

```
CREATE TABLE customer (
  customer_id SERIAL PRIMARY KEY,
  first_name VARCHAR(50),
  last_name  VARCHAR(50),
  email      VARCHAR(100),
  active     BOOLEAN
);
```

Values INSERT karo

```
INSERT INTO customer (first_name, last_name, email, active) VALUES
('Ravi',  'Kumar',  'ravi@gmail.com',        TRUE),
('Amit',  'Sharma', 'amit@ngo.org',          TRUE),
('Neha',  'Singh',  'neha@company.net',      TRUE),
('Pooja', 'Verma',  'pooja@yahoo.com',       TRUE),
('Karan', 'Mehta',  'karan@school.org',      FALSE),
('Ankit', 'Gupta',  'ankit@service.net',     TRUE);
```

data

```
SELECT * FROM customer;
```

query

```
SELECT customer_id, first_name, last_name, email
FROM customer
WHERE active = TRUE
 AND (
       email LIKE '%.org%'
    OR email LIKE '%.net%'
     );
```

Q4 — Active customers + email filter From customer, show: customer_id, first_name, last_name, email Find active customers whose email contains ".org" OR ".net".

CREATE karo

```
CREATE TABLE customer (
  customer_id SERIAL PRIMARY KEY,
  first_name VARCHAR(50),
  last_name  VARCHAR(50),
  email      VARCHAR(100),
  active     BOOLEAN
);
```

Values INSERT karo

```
INSERT INTO customer (first_name, last_name, email, active) VALUES
('Ravi',  'Kumar',  'ravi@gmail.com',     TRUE),
('Amit',  'Sharma', 'amit@ngo.org',       TRUE),
('Neha',  'Singh',  'neha@company.net',   TRUE),
('Pooja', 'Verma',  'pooja@yahoo.com',    TRUE),
('Karan', 'Mehta',  'karan@school.org',   FALSE),
('Ankit', 'Gupta',  'ankit@service.net',  TRUE);
```

Data check

```
SELECT * FROM customer;
```

query

```
SELECT customer_id, first_name, last_name, email
FROM customer
WHERE active = TRUE
 AND (email LIKE '%.org%' OR email LIKE '%.net%');
```

Q5 — Inactive customers in store 1 From customer, show: customer_id, store_id, active Find customers from store 1 who are not active. _____

CREATE karo

```
CREATE TABLE customer (
  customer_id SERIAL PRIMARY KEY,
  store_id INT,
  active BOOLEAN
);
```

Values INSERT karo

```
INSERT INTO customer (store_id, active) VALUES
(1, TRUE),
(1, FALSE),
(2, FALSE),
(1, FALSE),
(2, TRUE),
(1, TRUE);
```

Data check

```
SELECT * FROM customer;
```

query (FINAL)

```
SELECT customer_id, store_id, active
FROM customer
WHERE store_id = 1
 AND active = FALSE;
```

Q6 — Payment amount + date range From payment, show: payment_id, customer_id, amount, payment_date Find payments with amount between 2.00 and 5.00 and made in February 2007.

CREATE karo

```
CREATE TABLE payment (
  payment_id  SERIAL PRIMARY KEY,
  customer_id INT,
  amount      NUMERIC(5,2),
  payment_date TIMESTAMP
);
```

Values INSERT karo

```
INSERT INTO payment (customer_id, amount, payment_date) VALUES
(1, 1.99, '2007-02-10 10:15:00'),
(2, 2.50, '2007-02-05 12:30:00'),
(3, 4.00, '2007-02-18 09:00:00'),
(4, 5.00, '2007-02-25 20:10:00'),
(5, 6.50, '2007-02-12 14:00:00'),
(6, 3.00, '2007-03-02 11:00:00');
```

Data check

```
SELECT * FROM payment;
```

final query

```
SELECT payment_id, customer_id, amount, payment_date
FROM payment
WHERE amount BETWEEN 2.00 AND 5.00
 AND payment_date >= '2007-02-01'
 AND payment_date <  '2007-03-01';
```

Q7 — Rentals not returned From rental, show: rental_id, rental_date, return_date, customer_id Find rentals where return_date is NULL. _____

CREATE karo

```
CREATE TABLE rental (
  rental_id   SERIAL PRIMARY KEY,
  rental_date TIMESTAMP,
  return_date TIMESTAMP,
  customer_id INT
);
```

Values INSERT karo

```
INSERT INTO rental (rental_date, return_date, customer_id) VALUES
('2024-02-01 10:00:00', '2024-02-03 12:00:00', 1),
('2024-02-02 11:30:00', NULL,                  2),
('2024-02-03 09:15:00', '2024-02-05 18:00:00', 3),
('2024-02-04 14:00:00', NULL,                  4),
('2024-02-05 16:45:00', NULL,                  5);
```

Data check

```
SELECT * FROM rental;
```

query

```
SELECT rental_id, rental_date, return_date, customer_id
FROM rental
WHERE return_date IS NULL;
```

Q8 — Address district + postal code present From address, show: address_id, address, district, postal_code Find addresses where district is Texas or California AND postal_code is not NULL.

CREATE karo

```
CREATE TABLE address (
  address_id  SERIAL PRIMARY KEY,
  address     VARCHAR(150),
  district    VARCHAR(50),
  postal_code VARCHAR(20)
);
```

Values INSERT karo

```
INSERT INTO address (address, district, postal_code) VALUES
('101 Main Street',   'Texas',       '75001'),
('22 Park Avenue',   'California',  '90001'),
('55 Lake Road',     'Texas',       NULL),
('78 Hill Street',   'Florida',     '32003'),
('9 Sunset Blvd',    'California',  NULL),
('15 River Side',    'California',  '94016');
```

Data check

```
SELECT * FROM address;
```

query

```
SELECT address_id, address, district, postal_code
FROM address
WHERE (district = 'Texas' OR district = 'California')
 AND postal_code IS NOT NULL;
```

Q9 — Replacement cost + exclude titles From film, show: film_id, title, replacement_cost Find films where replacement_cost is 12.99, 16.99, or 28.99 AND the title does NOT contain the letter A.

CREATE karo

```
CREATE TABLE film (
  film_id SERIAL PRIMARY KEY,
  title VARCHAR(100),
  replacement_cost NUMERIC(5,2)
);
```

Values INSERT karo

```
INSERT INTO film (title, replacement_cost) VALUES
('Hero',        12.99),
('Matrix',      16.99),
('Rocky',       28.99),
('Superman',    12.99),
('Joker',       16.99),
('Titanic',     28.99),
('Glory',       10.99);
```

Data check

```
SELECT * FROM film;
```

query

```
SELECT film_id, title, replacement_cost
FROM film
WHERE replacement_cost IN (12.99, 16.99, 28.99)
 AND title NOT ILIKE '%a%';
```

Q10 — Inventory logic challenge From inventory, show: inventory_id, film_id, store_id Find inventory items where: • (store_id = 1 AND film_id between 1 and 50) OR • (store_id = 2 AND film_id between 51 and 100)

CREATE karo

```
CREATE TABLE inventory (
  inventory_id SERIAL PRIMARY KEY,
  film_id INT,
  store_id INT
);
```

Values INSERT karo

```
INSERT INTO inventory (film_id, store_id) VALUES
(10, 1),
(25, 1),
(55, 1),
(40, 2),
(60, 2),
(75, 2),
(90, 2),
(30, 1),
(51, 2),
(5,  2);
```

Data check

```
SELECT * FROM inventory;
```

query

```
SELECT inventory_id, film_id, store_id
FROM inventory
WHERE
    (store_id = 1 AND film_id BETWEEN 1 AND 50)
 OR (store_id = 2 AND film_id BETWEEN 51 AND 100);
```