```sql
-- Hello World SQL Query, SELECT all records FROM actor table
SELECT *
FROM actor;

-- Query for first_name and last_name in the actor table
SELECT first_name, last_name
FROM actor;

-- Query for first_name that equals Nick using WHERE clause
SELECT first_name,last_name
FROM actor
WHERE first_name = 'Nick';

-- Query for first_name that equals Nick using WHERE clause
-- using LIKE and WHERE
SELECT first_name,last_name
FROM actor
WHERE first_name LIKE 'Nick';

-- Query for all first_name data that starts with J
-- using LIKE WHERE and WILDCARD - %
SELECT first_name, actor_id
FROM actor
WHERE first_name LIKE 'J%';

-- Query for all first_name data that starts with K and has 2 letters
-- after it using LIKE WHERE and the underscore __
SELECT first_name, actor_id
FROM actor
WHERE first_name LIKE 'K__';

-- Query for all first_name data that starts with 'K' and ends with
'th'
-- using LIKE WHERE % (wildcard) and underscore _
SELECT first_name,last_name,actor_id
FROM actor
WHERE first_name LIKE 'K____%th';


-- Comparing Operators:
-----------------------
-- Greater Than (>) Less Than (<)
-- Greater or Equal (>=) Less or Equal (<=)
-- Not Equal (<>)

-- Explore the Payment Table
SELECT *
FROM payment;

-- Query for data that shows who paid an amount
-- GREATER than $2
SELECT customer_id, amount, payment_id
FROM payment
WHERE amount > 2.00;

-- Query for data showing who paid less than 7.99
```

```
SELECT customer_id, amount
FROM payment
WHERE amount < 7.99;

-- Query for data showing who paid less or equal to 7.99
SELECT customer_id, amount
FROM payment
WHERE amount <= 7.99;

-- Query for data showing who paid Greater or equal to $2
-- in Ascending ORDER -- ORDER BY defaults to Ascending
SELECT customer_id, amount
FROM payment
WHERE amount >= 2.00
ORDER BY amount;

-- Query for data showing who paid
-- an amount BETWEEN $2 and $8
-- using BETWEEN & AND clauses
SELECT customer_id, amount
FROM payment
WHERE amount BETWEEN 2.00 AND 7.99;

-- Query for data showing who paid
-- an amount not equal to $0.00
-- ORDER BY descending order
SELECT customer_id, amount
FROM payment
WHERE amount <> 0.00
ORDER BY amount DESC;


-- EXPLORE OTHER TABLES, AND COME BACK TO TELL US SOMETHING ABOUT
THEM.

-- Question
-- Query

-- SQL Aggregations => SUM(), AVG(), COUNT(), MIN(), MAX()

-- Display the sum of amounts payed that are greater than 5.99
SELECT SUM(amount)
FROM payment
WHERE amount > 5.99;

-- Display Average amounts of the same
SELECT AVG(amount)
FROM payment
WHERE amount > 5.99;

-- Display Count of amounts of the same
SELECT COUNT(amount)
FROM payment
WHERE amount > 5.99;

-- Display DISTINCT Count of amounts of the same
SELECT COUNT(DISTINCT amount)
```

```
FROM payment
WHERE amount > 5.99;

-- Display the min & max amount greater than 7.99
SELECT MIN(amount) AS min_num_payments, MAX(amount) AS max_num
FROM payment
WHERE amount >7.99;

-- GROUP BY is great with agregate functions

-- payments that are 7.99
SELECT amount, COUNT(amount)
FROM payment
WHERE amount = 7.99
GROUP BY amount;

-- How much did each customer pay in total?
SELECT customer_id, SUM(amount) AS customer_total
FROM payment
GROUP BY customer_id
ORDER BY customer_total DESC;

-- Group by customer_id and amount
SELECT customer_id, COUNT(amount) AS num_payments
FROM payment
GROUP BY customer_id
ORDER BY customer_id;

-- Check out Customer table
SELECT *
FROM customer;

-- Count the customers with "j" starting emails
SELECT COUNT(customer_id), email
FROM customer
WHERE email LIKE 'j%'
GROUP BY email
HAVING COUNT(customer_id) > 0;
```