1. **Film Table**

Check for duplicates:

```
SELECT title, film_id,
        COUNT(*)
FROM film
GROUP BY title, film_id
HAVING COUNT (*)>1
```

To clean the data, I would use CREATE VIEW -AS. Then I could use DELETE to get rid of any duplicates.

Check for unique data

```
SELECT DISTINCT *
FROM film
```

To clean this data, I would CREATE VIEW first. Then I would use UPDATE to change any misnamed or data that is not unique. To fix any missing values I would use SELECT and possibly exclude any columns with a few missing values. I can also UPDATE the table by imputing missing values using an average or other aggregate.

Customer Table

```
SELECT email, customer_id,address_id,
        COUNT(*)
FROM customer
GROUP BY email, customer_id, address_id
HAVING COUNT (*)>1
```

Check for unique data:

```
SELECT DISTINCT *
FROM customer
```

For both these, I would follow the same procedures as above.

2. **Film Table**

```
SELECT
        MIN(rental_duration) AS minimum_rental_duration,
        MAX(rental_duration) AS maximum_rental_duration,
        AVG(rental_duration) AS average_rental_duration,
        MIN(rental_rate) AS minimum_rental_rate,
        MAX(rental_rate)AS maximum_rental_rate,
        AVG(rental_rate)AS average_rental_rate,
        MIN(length)AS minimum_length,
        MAX(length)AS maxium_length,
```

```
    AVG(length) AS average_lenghth,
    MIN(replacement_cost) AS minimum_replacement_cost,
    MAX(replacement_cost) AS maxiuum_replacement_cost,
    AVG(replacement_cost) AS average_replacement_cost,
    MODE () WITHIN GROUP (ORDER BY rating) AS rating_mode,
    MODE() WITHIN GROUP (ORDER BY language_id) AS language_mode,
    MODE() WITHIN GROUP (ORDER BY release_year) AS year_mode
FROM film
```

Results:

minimum_rental_duration smallint	maximum_rental_duration smallint	average_rental_duration numeric
3	7	4.9850000000000000

minimum_rental_rate numeric	maximum_rental_rate numeric	average_rental_rate numeric
0.99	4.99	2.9800000000000000

minimum_length smallint	maxium_length smallint	average_lenghth numeric
46	185	115.2720000000000000

minimum_replacement_cost numeric	maxiuum_replacement_cost numeric	average_replacement_cost numeric
9.99	29.99	19.9840000000000000

rating_mode mpaa_rating	language_mode smallint	year_mode integer
PG-13	1	2006

Customer Table

```
SELECT
    MIN(customer_id) AS minimum_customer_id,
    MAX(customer_id) AS maximum_rental_customer_id,
    MIN(store_id) AS minimum_store_id,
    MAX(store_id)AS maximum_store_id,
    MODE() WITHIN GROUP (ORDER BY store_id) AS store_id_mode,
    MIN(address_id)AS minimum_address_id,
    MAX(address_id)AS maxium_address_id,
    MIN(create_date) AS minimum_create_date,
```

```
    MAX(create_date) AS maxiuum_create_date,
    MODE () WITHIN GROUP (ORDER BY create_date) AS rcreate_date_mode,
    MIN(last_update) AS minimum_last_update,
    MAX(last_update) AS maxiuum_last_update,
    MODE () WITHIN GROUP (ORDER BY last_update) AS last_update_mode,
    MODE() WITHIN GROUP (ORDER BY activebool) AS active_mode
FROM customer
```

minimum_customer_id integer	maximum_rental_customer_id integer	minimum_store_id smallint	maximum_store_id smallint
1	599	1	2

store_id_mode smallint	minimum_address_id smallint	maxium_address_id smallint	minimum_create_date date
1	5	605	2006-02-14

maxiuum_create_date date	rcreate_date_mode date	minimum_last_update timestamp without time zone	maxiuum_last_update timestamp without time z
2006-02-14	2006-02-14	2013-05-26 14:49:45.738	2013-05-26 14:49:45.738

maxiuum_last_update timestamp without time zone	last_update_mode timestamp without time zone	active_mode boolean
2013-05-26 14:49:45.738	2013-05-26 14:49:45.738	true

3.

SQL makes the data cleaning and profiling pretty easy. Once I get the 'grammar' it is much more intuitive than Excel. I do think Excel works well with smaller data sets and it does have some great graphic features that SQL does not have. Excel will also automatically find a lot of these descriptive statistics which is nice to have at your fingertips too.