1.

```sql
CREATE TABLE users (
  user_id SERIAL PRIMARY KEY,
  birth_date DATE,
  sex VARCHAR(6),
  age DECIMAL(2,0)
);

CREATE TABLE items (
  item_id SERIAL PRIMARY KEY,
  description VARCHAR,
  price FLOAT,
  category VARCHAR
);

CREATE TABLE rating (
  item_id INT,
  user_id INT,
  review VARCHAR
);
```

Типы данных:

- SERIAL – используется для автоматического задания растущих индексов в таблице.
- VARCHAR(6) – достаточный объем байтов для записи ‘male’ и ‘female’.
- DECIMAL (2,0) – int, но лишь двузначный, ограничение возраста для людей в таблице. Также можно использовать int, но условием в операторе CHECK.
- FLOAT – число с плавающей точкой; не int потому что нужен счет копеек + много работы с расчетом цен, ее лучше вести в float.
- VARCHAR – описание может быть сколь угодно длинным, а значит ограничивать данные нет необходимости.

Ключи:

- user_id – PK для users
- item_id – PK для items

```sql
ALTER TABLE rating
    ADD CONSTRAINT fk_rating_users FOREIGN KEY (user_id) REFERENCES users (user_id);

ALTER TABLE rating
    ADD CONSTRAINT fk_rating_items FOREIGN KEY (item_id) REFERENCES items (itebm_id);
```

```sql
ALTER TABLE rating
ADD COLUMN rating_id SERIAL

ALTER TABLE rating
ADD PRIMARY KEY (rating_id);
```

- fk_rating_users – FK для связи rating->users
- fk_rating_items - FK для связи rating->items
- rating_id – PK для rating (добавлен дополнительно – количество отзывов от одного человека и количество отзывов об одном товаре никак не ограничены)

Генерация значений:

- users

```sql
INSERT INTO users (
    user_id,
    birth_date,
    sex
  )
SELECT
  generate_series,
  now() - (random() * (interval '80 years')) - interval '20 years' AS random_date,
  ('[0:1]={female,male}'::text[])[trunc(random()*2)] AS random_choice
```

Возраст далее подтягивается отсюда:

```sql
UPDATE users
SET age = date_part('year',age(users.birth_date))
```

Вывод:

user_id	birth_date	age	sex
1	1943-08-05	80	female
2	1959-01-24	64	male
3	1993-08-27	30	male
4	1958-12-17	64	male
5	2000-11-03	22	male
6	1931-04-20	92	female
7	1998-12-15	24	male
8	1932-11-28	90	female
9	1949-12-12	73	male
10	1980-07-30	43	male
11	1948-06-24	75	female
12	1933-11-17	89	male
13	1990-05-11	33	male
14	1997-08-06	26	male
15	1938-04-02	85	female

- items

```
INSERT INTO items (
    item_id,
    description,
    price,
    category
  )
SELECT
  generate_series,
  md5(random()::text) || 'WB'  AS random_hash,
  FLOOR(random() * 1000)::int AS random_int,
  ('[0:3]={electronics, clothes, books, others}'::text[])[trunc(random()*4)] AS random_choice

FROM generate_series(1, 20)
```

Вывод:

item_id	description	price	category
1	92a13af34ba47dd05b183c…	86	books
2	17a30d1f56882fbe456536c…	180	books
3	cfb3873b546d9dffeede5e6…	11	books
4	28da7882a92efdcd788fbd2…	802	electronics
5	9c131c696c5e0553af550f9…	985	others
6	744bffb68b407404cf22fe67…	529	books
7	39ff34c88bf1712f9cbe50d1…	277	electronics
8	cd64e1e846da7a998df148…	513	others
9	dcf7416b01ab0fd2ee9ea9e…	125	electronics
10	d277661904d11f729735f21…	248	clothes
11	f87b8a610c8b9fd0a7b38fe…	207	electronics
12	cbc0f4e4daa8ea1555023f4…	893	books
13	50e8249e1425f9fea1ba92e…	13	electronics
14	7c9d16a83fdecbf7be9f68f2…	441	clothes
15	d62b93c4b880c8f89168c6…	784	electronics

- rating

```
INSERT INTO rating (
    rating_id,
    item_id,
    user_id,
    review
  )
SELECT
  generate_series,
  FLOOR(random() * 20 + 1)::int AS random_int_1,
  FLOOR(random() * 20 + 1)::int AS random_int_2,
  md5(random()::text) || 'WB' AS random_hash

FROM generate_series(1, 30)
```

Вывод:

item_id	user_id	review	rating_id
4	8	8dce52158b8eaa47460654...	1
11	9	26770c15bc93e0ff6d0988b...	2
8	2	4c0bc041a7798de40216f1...	3
15	3	0f1891e677f84a2e54603ea...	4
12	19	2bc3df7bf1247b82ce7fd26...	5
7	17	7671326f5625ac3f32aa8fe...	6
6	6	6e65cf459c5652880e27b4...	7
12	4	8e37e648bd6e56eb3a8bbb...	8
12	5	7e273541193d70cae35ced...	9
3	3	65666441cdbe1b71379bbc...	10
19	17	59322974d29567371a3bda...	11
19	20	d8028f97adad229362647d...	12
14	4	efdc8518bd8dccc5052351f...	13
2	20	15518c3ceef699892c1ef5b...	14
6	12	8d69c9fdf06166872ef45d8f...	15

2.

Между users и rating связь «один ко многим».

Между items и rating связь «один ко многим».

При этом связь между users и items – «многие ко многим» - несколько людей могут заказывать несколько товаров. Именно поэтому для ведения отзывов необходима «таблица-посредник», которая поддерживает связь с users и items с помощью FK.

3.

По-моему, имеет смысл вводить дополнительный индекс для таблиц users по столбцу age и items по столбцу category – при росте количества записей в них, индексы значительно ускорят запросы на чтение.

Для таблицы же rating индекс имеет смысл только если добавить в эту таблицу колонки «оценка товара» - тогда подтягивать данные о конкретном товаре в разрезе оценки станет эффективнее.

4.

1. INSERT INTO Car VALUES ('7984672834', 'E340BT, 77, Lada Granta', 'Красный', 87, 2017, 35)
   db error: ERROR: invalid input syntax for type integer: "Красный"

Красный передается в integer, что недопустимо. Даже после исправления кавычек, выпадает следующая ошибка:

db error: ERROR: insert or update on table "car" violates foreign key constraint "car_owner_fkey" DETAIL: Key (owner)=(7984672834) is not present in table "car_owner".

Это значит, что таблица car не может быть изменена, пока в car_owner нет записи с соответствующим ИНН.

2. Запрос выполнен. В таблицу car_owner добавлен владелец.

```
INSERT INTO Car_owner VALUES ('7984672834', 'Иван Петров')
```

inn	name
7984672834	Иван Петров

Теперь, если выполнить запрос 1 (исправив ошибки в воде данных), то выполнится и он:

```
INSERT INTO Car VALUES ('7984672834', 'E340BT', 77, 'Lada Granta',
 'Красный', 87, 2017, 35)
```

owner	curn	region	brand	color	power	caryear	mileage
7984672834	E340BT	77	Lada Granta	Красный	87	2017	35

3. db error: ERROR: duplicate key value violates unique constraint "car_owner_pkey" DETAIL: Key (inn)=(7984672834) already exists.

Так как поле inn является PK в таблице car_owner, то занесение еще одного владельца с тем же ИНН невозможно.

4. db error: ERROR: syntax error at or near "Owner"

Исправив опечатку (ее легко не заметить, так как OWNER – оператор в Postgres), получаем:

```
INSERT INTO Car_owner VALUES ('4752909757', 'Мван Петров')
```

inn	name
7984672834	Иван Петров
4752909757	Мван Петров

5. db error: ERROR: syntax error at or near "Volkswagen"

Исправим опечатку и получим:

db error: ERROR: duplicate key value violates unique constraint "car_pkey" DETAIL: Key (curn, region)=(E340BT, 77) already exists

В таблице car в качестве РК выступает совокупность столбцов curn, region. В таблице уже есть запись с таким РК, поэтому запись невозможна. Изменим регион на 75 и повторим попытку.

```
INSERT INTO Car VALUES ('6239572784', 'E340BT', 75, 'Volkswagen Polo',
'Синий', 105, 2018, 40)
```

db error: ERROR: insert or update on table "car" violates foreign key constraint "car_owner_fkey" DETAIL: Key (owner)=(6239572784) is not present in table "car_owner".

Аналогичная ошибка, что и в п.1.

6. Запрос выполнен:

```
INSERT INTO Car VALUES ('4752909757', 'A822EY', 99, 'Skoda Rapid',
'черный', 125, 2021, 35)
```

owner	curn	region	brand	color	power	caryear	mileage
7984672834	E340BT	77	Lada Granta	Красный	87	2017	35
4752909757	A822EY	99	Skoda Rapid	черный	125	2021	35

7. db error: ERROR: duplicate key value violates unique constraint "car_pkey" DETAIL: Key (curn, region)=(A822EY, 99) already exists.

В пункте 6 была создана запись с тем же РК, а значит запись невозможна.

8. db error: ERROR: syntax error at or near "Kia"

Снова опечатка, исправим:

```
INSERT INTO Car VALUES ('74478679847', '8971HP', 199, 'Kia Sportage',
'белый', 18, 2017, 35)
```

db error: ERROR: value too long for type character(10)

В ИНН 11 символов, хотя тип character(10) явно ограничивает количество байтов.

9. db error: ERROR: syntax error at or near "Toyota"

Опечатка, исправим:

```
INSERT INTO Car VALUES ('7984672834', 'E340BT', 77, 'Toyota RAV4',
'Серебристо-серый', 146, 2019)
```

И снова попытка внести авто в таблицу автомобилей с уже занесенным номером – конфликт PK.

Изменим регион на 27 и попробуем еще раз:

```
INSERT INTO Car VALUES ('7984672834', 'E340BT', 27, 'Toyota RAV4',
'Серебристо-серый', 146, 2019)
```

owner	curn	region	brand	color	power	caryear	mileage
7984672834	E340BT	77	Lada Granta	Красный	87	2017	35
4752909757	A822EY	99	Skoda Rapid	черный	125	2021	35
7984672834	E340BT	27	Toyota RAV4	Серебрис…	146	2019	NULL

Как видно, было зарегистрировано еще 1 авто. При этом, на вход подавалось на 1 параметр меньше, чем до этого – отсутствует «километраж». Но так как в описании таблицы это поле может быть пустым, то конфликта не возникло.

10. db error: ERROR: new row for relation "car" violates check constraint "car_power_check" DETAIL: Failing row contains (7984672834, H454EE, 98, Skoda Rapid, черный, 45, 2021, 0).

В таблицу car не допускается запись авто, у которых менее 50 л. с.. Это обеспечивается оператором CHECK в описании таблицы. Если увеличить мощность авто до 51, то получим следующее:

owner	curn	region	brand	color	power	caryear	mileage
7984672834	E340BT	77	Lada Granta	Красный	87	2017	35
4752909757	A822EY	99	Skoda Rapid	черный	125	2021	35
7984672834	E340BT	27	Toyota RAV4	Серебрис…	146	2019	NULL
7984672834	H454EE	98	Skoda Rapid	черный	51	2021	0

В таблицу добавлена новая запись.

5.

Допустим в таблице Documents N строк. Тогда, данный скрипт

```sql
INSERT INTO docs00
SELECT * FROM documents WHERE (id%16)=0

...

INSERT INTO docs15
SELECT * FROM documents WHERE (id%16)=15
```

разделит таблицу на 16 подтаблиц (шардов), каждая из которых будет хранить N/16 строк. Если допустить, что в данном запросе каждый из доков лежит на отдельном сервере, то такое деление будет являться горизонтальным шардированием.

Другой вопрос – получится ли каждая из полученных подтаблиц тестовой?
Если N достаточно велико, то распределение данных будет относительно равномерно.

Однако, чаще логичней вводить шардирование по более очевидному признаку – например, дате.