University of Science – Vietnam National University
Faculty of Information Technology

CS423 – Software Testing

HW03 Report

Data Generation

Student's Name	Student ID	Class
Tran Nhat Thanh	22125093	22TT

Ho Chi Minh City, November 25th, 2025

Contents

1. Introduction

The goal of this task is to generate a large-scale, relational dataset for the **Toolshop** application. Given the need for over 2,000 records with strict referential integrity (linking valid users, products, and transactions), manual entry was impractical.

I developed a custom **Python automation tool** to programmatically generate this data, ensuring accuracy and realism. The tool utilizes:

- **Faker**: For realistic synthetic data (names, addresses, identifiers, etc.).
- **openpyxl**: To generate professional Excel files with formatted headers and auto-sized columns.
- **random**: For applying business logic distributions (cart sizes).

Unlike simple generators, this script enforces critical business rules, such as dynamic invoice calculation and strict phone number formatting for spreadsheet compatibility.

2. Data Categories and Field Specifications

The tool generates data for four relational entities. The schema below has been adapted to align with the **original Toolshop database (SQL)** structure, with minor adjustments for reporting clarity.

A. Categories (50 Records)

Field	Data Type	Generation Logic / Range
id	Integer	Sequential (1 to 50).
parent_id	Integer	Links sub-categories ("Hammer") to parent categories ("Hand Tools").
name	String	Selected from a curated list of 54 industry-specific tool names ("Cordless Drill", "Generator", etc.).
slug	String	URL-friendly version of the name (lowercase, spaces replaced with hyphens).

B. Users (50 Records)

Field	Data Type	Generation Logic / Range
id	Integer	Sequential (1 to 50).
first_name	String	Random first name.
last_name	String	Random last name.
address	String	Random street address.
city	String	Random city name.
state	String	Random state name.
country	String	Random real-world country name.
postcode	String	Random postal code.

phone	String	Formatted string: (###) ###-#### to prevent Excel formula errors.
dob	Date	YYYY-MM-DD to match SQL standard.
email	String	Unique random email address.
password	String	Hardcoded hash string (from SQL dump).
role	String	Defaults to “user”. (original SQL has “admin”)

C. Products (1000 Records)

Field	Data Type	Generation Logic / Range
id	Integer	Sequential (1 to 1000).
name	String	Composite string: Random Color + Category Name ("Red Hammer").
description	String	**Template-based:** "This {adjective} {product_name} is designed for {use_case}..." ensuring relevance to the item name.
stock	Integer	Random integer between 0 and 100.
price	Float	Random float between $5.00 and $200.00.
brand_id	Integer	Random integer 1–10 to match SQL foreign keys.
category_id	Integer	Foreign Key: Random integer 1 to 50.
product_image_id	String	Generated ULID (01J...) to simulate valid image references.
is_location_offer	Boolean	0 or 1.
is_rental	Boolean	0 or 1.
co2_rating	String	Restricted list: "A" through "E". Added meaning.

D. Transactions (1000 Records)

Field	Data Type	Generation Logic / Range
id	Integer	Sequential (1 to 1000).
user_id	Integer	Foreign Key: Random integer 1 to 50.
invoice_date	DateTime	Random timestamp (past 2 years). Format: YYYY-MM-DD HH:MM:SS.
invoice_number	String	Pattern: INV-{Year}{ID}.
billing_address	String	Pulled dynamically from the linked User.
billing_city	String	Pulled dynamically from the linked User.
billing_state	String	Pulled dynamically from the linked User.
billing_postcode	String	Pulled dynamically from the linked User.
total	Float	Calculated Sum: Sum of prices of items in purchased_items.

payment_method	String	Random choice (Credit Card, Bank Transfer, etc.).
payment_account_name	String	Matches User's full name.
payment_account_number	String	Random alphanumeric string.
created_at	DateTime	Matches invoice_date.
status	String	Restricted list: AWAITING_FULFILLMENT, SHIPPED, etc.
purchased_items	JSON	JSON array of items to preserve purchase history in CSV format.

Note: The data follows the production SQL schema, with two added fields for standalone validation: a status field that mimics API-visible states not stored in the base invoices table, and a denormalized purchased_items JSON column that embeds many-to-many line-item data, allowing total verification without a join table.

3. Tool Execution & Workflow

The tool requires **Python 3.x** and two libraries: faker (for data generation) and openpyxl (for Excel styling).

Step 1: Library Installation

The dependencies were installed via pip:

pip install faker openpyxl

Step 2: Running the Script

The script creates all four files in a single execution pass.

```
D:\HCMUS\4.Senior\1stSemester\CS423\Assignments\Homework\HW03>python -u "d:\HCMUS\4.Senior\1stSemester\CS423\
Assignments\Homework\HW03\toolshop_data_generator.py"
Saving 50 rows to categories.xlsx...
Saving 50 rows to users.xlsx...
Saving 1000 rows to products.xlsx...
Saving 1000 rows to transactions.xlsx...

SUCCESS! 4 data files generated successfully.
```

4. Source Code Explanation

The following section explains the core logic of the custom tool, broken down by function.

A. Configuration and Setup

I utilized the Faker library to generate realistic synthetic data and initialized three global caches – product_cache, user_cache, and category_cache – to store generated data in memory. This multi-level caching strategy enables the script to maintain strict semantic integrity across all datasets (for example: ensuring a "Hammer" product is linked to the "Hand Tools" category rather than "Rentals").

Additionally, I defined static constant lists (BASE_CATEGORIES, BRANDS, etc.) to enforce business logic. Notably, the BASE_CATEGORIES list was initially generated using AI to ensure a diverse and realistic set of 54 industry-specific tool names, which were then hardcoded into the script to guarantee correctness and consistency with the Toolshop domain.

```python
# CONFIGURATION
fake = Faker()
Faker.seed(12345)

# GLOBAL CACHES
category_cache = []
product_cache = []
user_cache = []

# CONSTANTS
CO2_RATINGS = [
    "None", "A (Lowest Impact)", "B (Low Impact)",
    "C (Moderate Impact)", "D (Higher Impact)", "E (Highest Impact)"
]

TRANSACTION_STATUSES = [
    "AWAITING_FULFILLMENT", "ON_HOLD",
    "AWAITING_SHIPMENT", "SHIPPED", "COMPLETED"
]

PAYMENT_METHODS = ["Cash on Delivery", "Credit Card", "Bank Transfer", "Gift Card", "Buy Now Pay Later"]

CATEGORY_STRUCTURE = {
    "Hand Tools": [
        "Hammer", "Claw Hammer", "Mallet", "Sledgehammer",
        "Wood Saw", "Hand Saw", "Hacksaw", "Chisel", "File",
        "Adjustable wrench", "Wrench", "Pipe Wrench", "Torque Wrench",
        "Open-end spanners (Set)", "Phillips Screwdriver", "Screwdriver",
        "Pliers", "Combination Pliers", "Bolt Cutters", "Long Nose Pliers", "Slip Joint Pliers",
        "Utility Knife", "Tape Measure", "Level"
    ],
    "Power Tools": [
        "Sheet Sander", "Belt Sander", "Random Orbit Sander", "Sander",
        "Cordless Drill", "Cordless Drill 18V", "Drill Bits", "Drill",
        "Grinder", "Angle Grinder", "Circular Saw", "Jigsaw", "Reciprocating Saw", "Saw",
        "Heat Gun", "Router", "Planer"
    ],
    "Rentals": [
        "Crane", "Excavator", "Bulldozer", "Jackhammer",
        "Concrete Mixer", "Generator", "Welding Machine", "Air Compressor", "Pressure Washer"
    ],
    "Other": [
```

B. Excel Formatting Logic

This helper function is responsible for creating the physical Excel files. Instead of standard CSV writing, it uses *openpyxl* to apply professional formatting that standard text files cannot support, making the output files immediately readable.

```
61  # SAVE FUNCTION
62  def save_as_excel(filename, headers, data_rows):
63      print(f"Saving {len(data_rows)} rows to {filename}...")
64      wb = Workbook()
65      ws = wb.active
66      ws.append(headers)
67
68      for cell in ws[1]:
69          cell.font = Font(bold=True)
70
71      for row in data_rows:
72          ws.append(row)
73
74      for column_cells in ws.columns:
75          max_length = 0
76          column = column_cells[0].column_letter
77          for cell in column_cells:
78              try:
79                  if len(str(cell.value)) > max_length:
80                      max_length = len(str(cell.value))
81              except: pass
82          adjusted_width = (max_length + 2)
83          ws.column_dimensions[column].width = adjusted_width
84
85      wb.save(filename)
```

C. Generating Categories

This function aligns with the SQL schema by implementing a **Parent-Child relationship**. It first generates root categories ("Hand Tools") and then generates specific tools ("Hammer") linked to those parents via parent_id. This ensures the hierarchical structure of the product catalog is accurately represented.

```python
def generate_categories(count=50):
    global category_cache
    headers = ["id", "parent_id", "name", "slug"]
    rows = []

    current_id = 1

    parents = {}
    for parent_name in CATEGORY_STRUCTURE.keys():
        slug = parent_name.lower().replace(" ", "-")
        rows.append([current_id, None, parent_name, slug])
        parents[parent_name] = current_id

        category_cache.append({"id": current_id, "name": parent_name})
        current_id += 1

    all_children = []
    for p_name, children in CATEGORY_STRUCTURE.items():
        for c_name in children:
            all_children.append((p_name, c_name))

    remaining_slots = count - len(parents)

    for i in range(remaining_slots):
        if i < len(all_children):
            parent_name, child_name = all_children[i]
            parent_id = parents[parent_name]
            name = child_name
        else:
            name = f"Specialty Tool {i}"
            parent_id = parents["Other"]

        slug = name.lower().replace(" ", "-")

        rows.append([current_id, parent_id, name, slug])

        # Cache for products
        category_cache.append({"id": current_id, "name": name})
        current_id += 1

    save_as_excel("categories.xlsx", headers, rows)
```

D. Generating Users

This function generates user profiles matching the SQL schema. It creates a global user_cache dictionary containing the user's ID and full address, which is later used to populate billing details in the Transaction file.

```python
def generate_users(count=50):
    global user_cache
    headers = ["id", "first_name", "last_name", "address", "city", "state",
               "country", "postcode", "phone", "dob", "email", "password", "role"]
    rows = []

    for i in range(1, count + 1):
        fname = fake.first_name()
        lname = fake.last_name()
        addr = fake.street_address()
        city = fake.city()
        state = fake.state()
        country = fake.country()
        postcode = fake.postcode()
        dob_sql = fake.date_of_birth(minimum_age=18, maximum_age=70).strftime("%Y-%m-%d")

        clean_phone = fake.numerify('(###) ###-####')

        # Cache for transactions
        user_cache.append({
            "id": i,
            "name": f"{fname} {lname}",
            "address": addr,
            "city": city,
            "state": state,
            "country": country,
            "postcode": postcode
        })

        rows.append([
            i, fname, lname, addr, city, state, country, postcode,
            clean_phone, dob_sql, fake.email(),
            "9e2ed9cb4bf54a6b9dc4669a1d295466b2585c4346092bffb5333098431cd61d", # Hash from SQL
            "user"
        ])

    save_as_excel("users.xlsx", headers, rows)
```

E. Generating Products

This function builds the inventory. To ensure semantic validity, it looks up a real Category from the cache ("Hammer") and generates a product name based on that category ("Blue Hammer"). It also links to brand_id (integer) and generates a ULID for product_image_id.

```python
def generate_products(count=1000):
    global product_cache
    if not category_cache:
        print("Error: Category cache empty.")
        return

    headers = ["id", "name", "description", "stock", "price", "brand_id",
               "category_id", "product_image_id", "is_location_offer", "is_rental", "co2_rating"]
    rows = []

    ADJECTIVES = ["durable", "lightweight", "heavy-duty", "ergonomic", "precision-engineered",
                  "compact", "versatile", "high-performance", "reliable", "industry-standard"]

    USE_CASES = ["professional construction", "home DIY projects", "industrial applications",
                 "precision tasks", "heavy lifting", "everyday repairs"]

    for i in range(1, count + 1):
        cat_data = random.choice(category_cache)
        cat_id = cat_data['id']
        cat_name = cat_data['name']

        name = f"{fake.color_name().capitalize()} {cat_name}"
        price = round(random.uniform(5.00, 200.00), 2)

        adj = random.choice(ADJECTIVES)
        use = random.choice(USE_CASES)
        description = f"This {adj} {name} is designed for {use}. It features a robust build quality ensuring long-lasting performan

        mock_image_id = "01J" + fake.bothify(text='?#?#?#?#?#?#?#?#?#?#?#?#').upper()

        product_cache.append({
            "id": i,
            "name": name,
            "price": price
        })

        rows.append([
            i, name, description,
            random.randint(0, 100),
            price,
            random.randint(1, 10),
            cat_id,
            mock_image_id,
            random.choice([0, 1]),
            random.choice([0, 1]),
            random.choice(CO2_RATINGS)
        ])

    save_as_excel("products.xlsx", headers, rows)
```

F. Generating Transactions

This function simulates realistic purchasing behavior. It retrieves a real User from user_cache to populate the billing fields (ensuring the invoice address matches the user). It then selects random items from product_cache, calculates the true total, and saves the item list as JSON.

```python
def generate_transactions(count=1000):
    if not product_cache or not user_cache:
        print("Error: Caches empty. Script execution order is wrong.")
        return

    headers = ["id", "user_id", "invoice_date", "invoice_number",
               "billing_address", "billing_city", "billing_state", "billing_country", "billing_postcode",
               "total", "payment_method", "payment_account_name", "payment_account_number",
               "created_at", "status", "purchased_items"]
    rows = []

    for i in range(1, count + 1):
        real_user = random.choice(user_cache)

        num_items = random.randint(1, 5)
        cart_items = random.sample(product_cache, num_items)
        cart_total = sum(item['price'] for item in cart_items)
        items_json = json.dumps(cart_items)

        inv_date = fake.date_time_between(start_date='-2y', end_date='now')
        inv_date_str = inv_date.strftime("%Y-%m-%d %H:%M:%S")

        rows.append([
            i,
            real_user['id'],
            inv_date_str,
            f"INV-{inv_date.year}{i:08d}",
            real_user['address'],
            real_user['city'],
            real_user['state'],
            real_user['country'],
            real_user['postcode'],
            round(cart_total, 2),
            random.choice(PAYMENT_METHODS),
            real_user['name'],
            fake.bothify(text='#########???').upper(),
            inv_date_str,
            random.choice(TRANSACTION_STATUSES),
            items_json
        ])

    save_as_excel("transactions.xlsx", headers, rows)
```

5. Sample Data Screenshots

Below are previews of the generated datasets viewed in Excel. Note that due to the high dimensionality of the data, the screenshots serve only as a partial preview. For full details, please consult the actual data files.

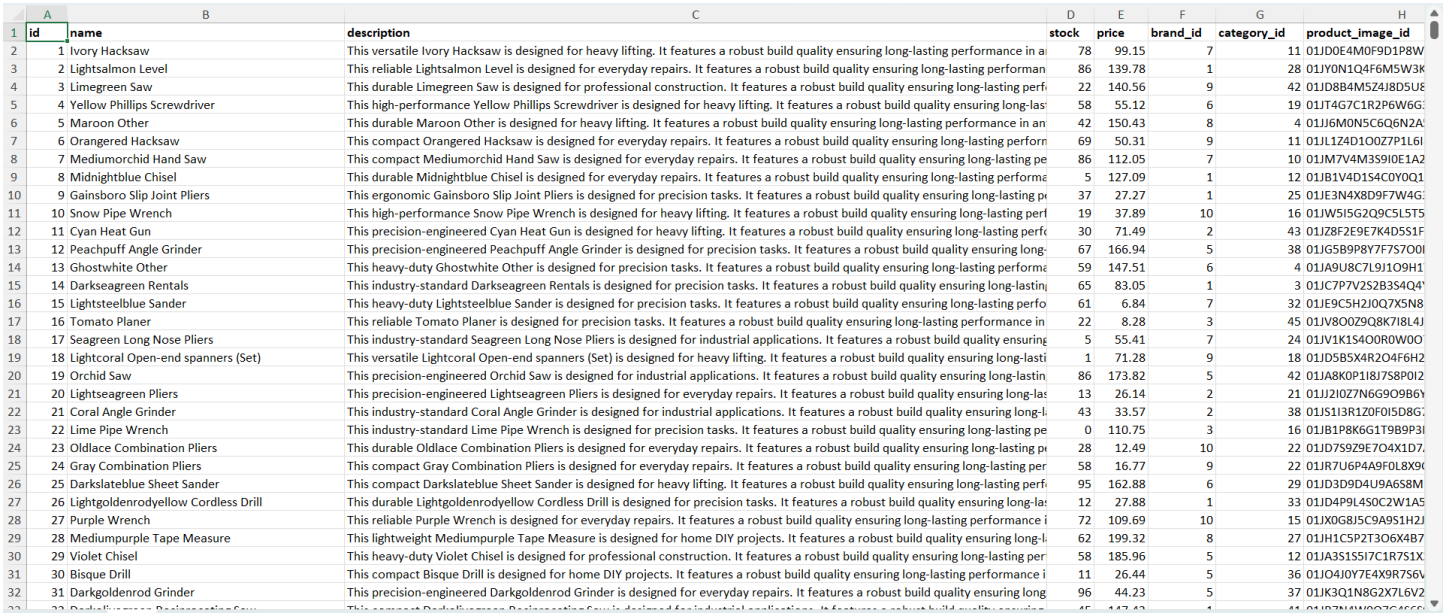
A. Categories Data

	A	B	C	D
1	**id**	**parent_id**	**name**	**slug**
2	1		Hand Tools	hand-tools
3	2		Power Tools	power-tools
4	3		Rentals	rentals
5	4		Other	other
6	5	1	Hammer	hammer
7	6	1	Claw Hammer	claw-hammer
8	7	1	Mallet	mallet
9	8	1	Sledgehammer	sledgehammer
10	9	1	Wood Saw	wood-saw
11	10	1	Hand Saw	hand-saw
12	11	1	Hacksaw	hacksaw
13	12	1	Chisel	chisel
14	13	1	File	file
15	14	1	Adjustable wrench	adjustable-wrench
16	15	1	Wrench	wrench
17	16	1	Pipe Wrench	pipe-wrench
18	17	1	Torque Wrench	torque-wrench
19	18	1	Open-end spanners (Set)	open-end-spanners-(set)
20	19	1	Phillips Screwdriver	phillips-screwdriver
21	20	1	Screwdriver	screwdriver

B. Users Data

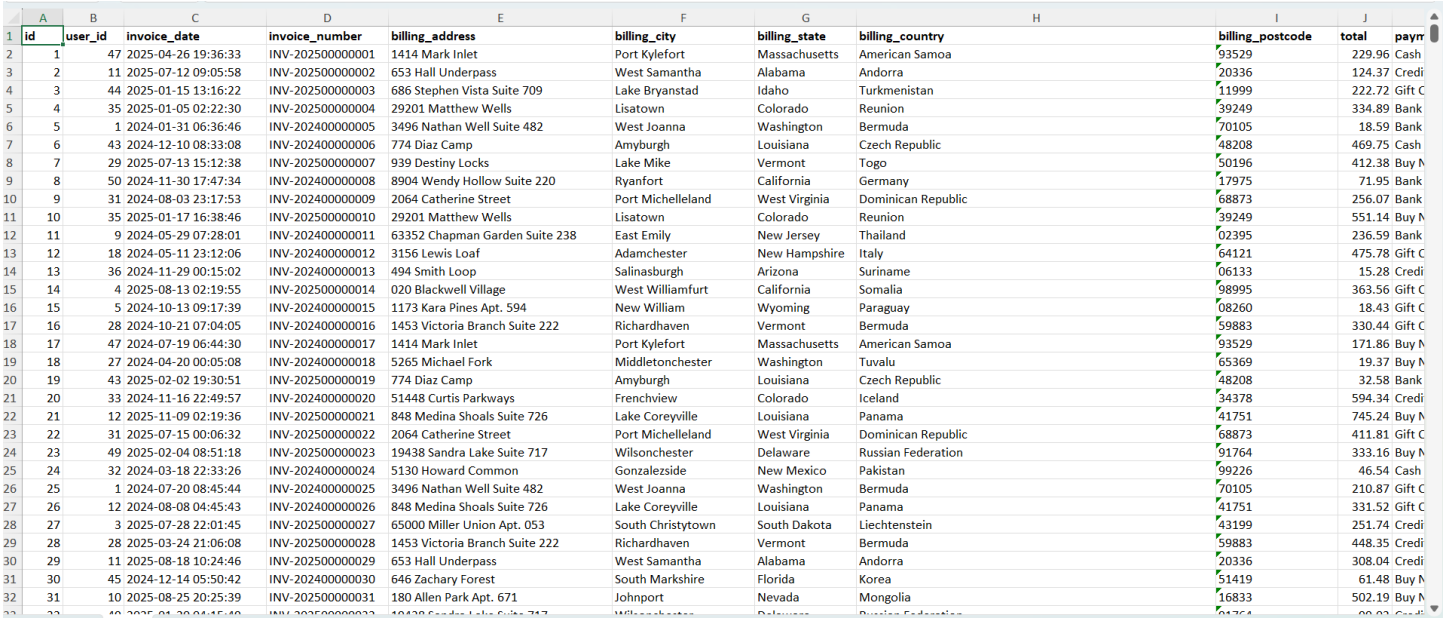
	A	B	C	D	E	F	G	H	I	J	K
1	**id**	**first_name**	**last_name**	**address**	**city**	**state**	**country**	**postcode**	**phone**	**dob**	**email**
2	1	Robert	Smith	3496 Nathan Well Suite 482	West Joanna	Washington	Bermuda	70105	(698) 223-1355	2004-12-11	ellisanthony@exam
3	2	Tiffany	Garner	29241 Rodriguez Springs	Lake Nicholas	North Dakota	Nepal	93481	(031) 738-2496	1977-05-08	bobmartin@exampl
4	3	Edward	Perez	65000 Miller Union Apt. 053	South Christytown	South Dakota	Liechtenstein	43199	(968) 351-8217	2007-06-08	patriciarogers@exa
5	4	Frank	Roberts	020 Blackwell Village	West Williamfurt	California	Somalia	98995	(623) 205-5262	1986-04-06	ashleycolon@exam
6	5	Susan	Palmer	1173 Kara Pines Apt. 594	New William	Wyoming	Paraguay	08260	(605) 889-1125	1972-08-03	hillscott@example.c
7	6	Richard	Cole	7704 Johnson Key Apt. 010	New Monicafurt	New York	Cocos (Keeling) Islands	36234	(345) 619-4506	1978-08-29	ethanthomas@exan
8	7	Jesse	Coleman	18056 Delacruz Garden	South Eddieborough	Kentucky	Montenegro	70278	(240) 118-3886	1979-04-29	stricklandedward@e
9	8	Mark	Farmer	6705 Miranda Knoll	West Thomaston	Massachusetts	Germany	51550	(269) 909-1711	1975-03-19	arthurwilliams@exa
10	9	Alicia	Stevens	63352 Chapman Garden Suite 238	East Emily	New Jersey	Thailand	02395	(311) 811-4933	1999-03-05	jeremiah57@examp
11	10	Richard	Wright	180 Allen Park Apt. 671	Johnport	Nevada	Mongolia	16833	(396) 492-0840	1966-07-21	sroach@example.ne
12	11	James	Garrett	653 Hall Underpass	West Samantha	Alabama	Andorra	20336	(925) 329-9484	1993-03-30	clarkphillip@exampl
13	12	Bianca	Gibbs	848 Medina Shoals Suite 726	Lake Coreyville	Louisiana	Panama	41751	(671) 616-3907	1982-12-27	courtneykelly@exan
14	13	Joseph	Thomas	5956 Michael Spurs	Johnsonfurt	New York	Burundi	80068	(594) 827-7075	2007-02-14	robertruiz@example
15	14	William	Vasquez	5122 Kathleen Spur	Port Paulstad	California	Cook Islands	86887	(015) 493-8949	1969-07-15	michael60@exampl
16	15	David	Kim	41240 Donald Flat Suite 345	North Sharonberg	West Virginia	Djibouti	93786	(200) 266-9602	1957-12-09	emilyblack@exampl
17	16	Lori	Kelly	201 Anderson Lake	Susanport	Illinois	Mauritius	43956	(152) 478-2826	1982-07-07	wallacenicole@exar
18	17	Derek	Sutton	440 Dawson Plaza Apt. 611	South Linda	Idaho	Belarus	87305	(116) 207-1644	1994-10-28	alexandriaperez@ex
19	18	Gregory	Garcia	3156 Lewis Loaf	Adamchester	New Hampshire	Italy	64121	(773) 905-2833	1962-01-21	bergervalerie@exan
20	19	Lisa	Ballard	22012 Rhonda Throughway	West Davidside	Maryland	Zimbabwe	02201	(993) 510-0614	1987-09-02	ecantrell@example.
21	20	Stephanie	Hayes	286 Robert Divide Suite 536	South Mauricefort	Mississippi	British Indian Ocean Territory (Chagos Archipelago)	80247	(183) 016-5872	1973-07-01	karapatterson@exa
22	21	David	James	60397 Shaw Port	Fisherchester	Arkansas	Colombia	86495	(157) 040-7818	2003-05-23	ihamilton@example
23	22	Christopher	Shepherd	705 Murphy Junctions Suite 887	Cynthiachester	Colorado	Christmas Island	33086	(798) 154-8575	1988-03-05	hannah85@example
24	23	Hunter	Jones	7793 Walker Track Apt. 575	Cardenashaven	Connecticut	Solomon Islands	03712	(570) 302-5903	2003-04-14	carterrandy@examp
25	24	Deborah	Gonzalez	5633 Dyer Alley Apt. 393	Doyleton	Virginia	Taiwan	61645	(278) 593-5012	1964-05-04	monicahampton@e:
26	25	Mark	Hamilton	853 Miller Flat Suite 312	New Nathan	Delaware	Norway	36535	(538) 970-4475	1964-08-09	andrademichelle@e
27	26	Dominique	Johnson	301 Kayla Lake Suite 770	Lake Kristinhaven	California	Mongolia	74587	(539) 658-0918	1961-03-05	christineturner@exa
28	27	Robert	Tanner	5265 Michael Fork	Middletonchester	Washington	Tuvalu	65369	(436) 136-2898	2004-07-30	obrienjessica@exam
29	28	Alexander	Edwards	1453 Victoria Branch Suite 222	Richardhaven	Vermont	Bermuda	59883	(156) 199-6118	1995-04-07	sarahmosley@exam
30	29	Lee	Holden	939 Destiny Locks	Lake Mike	Vermont	Togo	50196	(410) 557-1403	2003-10-21	nicholas04@exampl
31	30	Selena	Smith	870 Stephanie Pine Apt. 641	Jillland	Pennsylvania	Yemen	45756	(158) 098-5036	1983-04-03	stewartjoseph@exa
32	31	Kristine	English	2064 Catherine Street	Port Michelleland	West Virginia	Dominican Republic	68873	(606) 864-1011	1973-04-12	halledward@examp

C. Products Data



id	name	description	stock	price	brand_id	category_id	product_image_id
1	Ivory Hacksaw	This versatile Ivory Hacksaw is designed for heavy lifting. It features a robust build quality ensuring long-lasting performance in a	78	99.15	7	11	01JD0E4M0F9D1P8W
2	Lightsalmon Level	This reliable Lightsalmon Level is designed for everyday repairs. It features a robust build quality ensuring long-lasting performan	86	139.78	1	28	01JY0N1Q4F6M5W3K
3	Limegreen Saw	This durable Limegreen Saw is designed for professional construction. It features a robust build quality ensuring long-lasting perf	22	140.56	9	42	01JD8B4M5Z4J8D5U8
4	Yellow Phillips Screwdriver	This high-performance Yellow Phillips Screwdriver is designed for heavy lifting. It features a robust build quality ensuring long-las	58	55.12	6	19	01JT4G7C1R2P6W6G3
5	Maroon Other	This durable Maroon Other is designed for heavy lifting. It features a robust build quality ensuring long-lasting performance in an	42	150.43	8	4	01JJ6M0N5C6Q6N2A5
6	Orangered Hacksaw	This compact Orangered Hacksaw is designed for everyday repairs. It features a robust build quality ensuring long-lasting perforn	69	50.31	9	11	01JL1Z4D1O0Z7P1L6I
7	Mediumorchid Hand Saw	This compact Mediumorchid Hand Saw is designed for everyday repairs. It features a robust build quality ensuring long-lasting pe	86	112.05	7	10	01JM7V4M3S9I0E1A2
8	Midnightblue Chisel	This durable Midnightblue Chisel is designed for everyday repairs. It features a robust build quality ensuring long-lasting performa	5	127.09	1	12	01JB1V4D1S4C0Y0Q1
9	Gainsboro Slip Joint Pliers	This ergonomic Gainsboro Slip Joint Pliers is designed for precision tasks. It features a robust build quality ensuring long-lasting pe	37	27.27	1	25	01JE3N4X8D9F7W4G3
10	Snow Pipe Wrench	This high-performance Snow Pipe Wrench is designed for heavy lifting. It features a robust build quality ensuring long-lasting perf	19	37.89	10	16	01JW5I5G2Q9C5L5T5
11	Cyan Heat Gun	This precision-engineered Cyan Heat Gun is designed for heavy lifting. It features a robust build quality ensuring long-lasting perfo	30	71.49	2	43	01JZ8F2E9E7K4D5S1F
12	Peachpuff Angle Grinder	This precision-engineered Peachpuff Angle Grinder is designed for precision tasks. It features a robust build quality ensuring long-	67	166.94	5	38	01JG5B9P8Y7F7S7O0I
13	Ghostwhite Other	This heavy-duty Ghostwhite Other is designed for precision tasks. It features a robust build quality ensuring long-lasting performa	59	147.51	6	4	01JA9U8C7L9J1O9H1
14	Darkseagreen Rentals	This industry-standard Darkseagreen Rentals is designed for precision tasks. It features a robust build quality ensuring long-lastin	65	83.05	1	3	01JC7P7V2S2B3S4Q4Y
15	Lightsteelblue Sander	This heavy-duty Lightsteelblue Sander is designed for precision tasks. It features a robust build quality ensuring long-lasting perfo	61	6.84	7	32	01JE9C5H2J0Q7X5N8
16	Tomato Planer	This reliable Tomato Planer is designed for precision tasks. It features a robust build quality ensuring long-lasting performance in	22	8.28	3	45	01JV8O0Z9Q8K7I8L4J
17	Seagreen Long Nose Pliers	This industry-standard Seagreen Long Nose Pliers is designed for industrial applications. It features a robust build quality ensurin	5	55.41	7	24	01JV1K1S4O0R0W0O7
18	Lightcoral Open-end spanners (Set)	This versatile Lightcoral Open-end spanners (Set) is designed for heavy lifting. It features a robust build quality ensuring long-lasti	1	71.28	9	18	01JD5B5X4R2O4F6H2
19	Orchid Saw	This precision-engineered Orchid Saw is designed for industrial applications. It features a robust build quality ensuring long-lastin	86	173.82	5	42	01JA8K0P1I8J7S8P0I2
20	Lightseagreen Pliers	This precision-engineered Lightseagreen Pliers is designed for everyday repairs. It features a robust build quality ensuring long-las	13	26.14	2	21	01JJ2I0Z7N6G9O9B6Y
21	Coral Angle Grinder	This industry-standard Coral Angle Grinder is designed for industrial applications. It features a robust build quality ensuring long-l	43	33.57	2	38	01JS1I3R1Z0F0I5D8G7
22	Lime Pipe Wrench	This industry-standard Lime Pipe Wrench is designed for precision tasks. It features a robust build quality ensuring long-lasting pe	0	110.75	3	16	01JB1P8K6G1T9B9P3I
23	Oldlace Combination Pliers	This durable Oldlace Combination Pliers is designed for everyday repairs. It features a robust build quality ensuring long-lasting pe	28	12.49	10	22	01JD7S9Z9E7O4X1D7A
24	Gray Combination Pliers	This compact Gray Combination Pliers is designed for everyday repairs. It features a robust build quality ensuring long-lasting per	58	16.77	9	22	01JR7U6P4A9F0L8X9C
25	Darkslateblue Sheet Sander	This compact Darkslateblue Sheet Sander is designed for heavy lifting. It features a robust build quality ensuring long-lasting perfo	95	162.88	6	29	01JD3D9D4U9A6S8M1
26	Lightgoldenrodyellow Cordless Drill	This durable Lightgoldenrodyellow Cordless Drill is designed for precision tasks. It features a robust build quality ensuring long-las	12	27.88	1	33	01JD4P9L4S0C2W1A5
27	Purple Wrench	This reliable Purple Wrench is designed for everyday repairs. It features a robust build quality ensuring long-lasting performance i	72	109.69	10	15	01JX0G8J5C9A9S1H2J
28	Mediumpurple Tape Measure	This lightweight Mediumpurple Tape Measure is designed for home DIY projects. It features a robust build quality ensuring long-l	62	199.32	8	27	01JH1C5P2T3O6X4B7
29	Violet Chisel	This heavy-duty Violet Chisel is designed for professional construction. It features a robust build quality ensuring long-lasting per	58	185.96	5	12	01JA3S1S5I7C1R7S1X
30	Bisque Drill	This compact Bisque Drill is designed for home DIY projects. It features a robust build quality ensuring long-lasting performance i	11	26.44	5	36	01JO4J0Y7E4X9R7S6V
31	Darkgoldenrod Grinder	This precision-engineered Darkgoldenrod Grinder is designed for everyday repairs. It features a robust build quality ensuring long	96	44.23	5	37	01JK3Q1N8G2X7L6V2

D. Transactions Data



id	user_id	invoice_date	invoice_number	billing_address	billing_city	billing_state	billing_country	billing_postcode	total	payn
1	47	2025-04-26 19:36:33	INV-202500000001	1414 Mark Inlet	Port Kylefort	Massachusetts	American Samoa	93529	229.96	Cash
2	11	2025-07-12 09:05:58	INV-202500000002	653 Hall Underpass	West Samantha	Alabama	Andorra	20336	124.37	Credi
3	44	2025-01-15 13:16:22	INV-202500000003	686 Stephen Vista Suite 709	Lake Bryanstad	Idaho	Turkmenistan	11999	222.72	Gift C
4	35	2025-01-05 02:22:30	INV-202500000004	29201 Matthew Wells	Lisatown	Colorado	Reunion	39249	334.89	Bank
5	1	2024-01-31 06:36:46	INV-202400000005	3496 Nathan Well Suite 482	West Joanna	Washington	Bermuda	70105	18.59	Bank
6	43	2024-12-10 08:33:08	INV-202400000006	774 Diaz Camp	Amyburgh	Louisiana	Czech Republic	48208	469.75	Cash
7	29	2025-07-13 15:12:38	INV-202500000007	939 Destiny Locks	Lake Mike	Vermont	Togo	50196	412.38	Buy N
8	50	2024-11-30 17:47:34	INV-202400000008	8904 Wendy Hollow Suite 220	Ryanfort	California	Germany	17975	71.95	Bank
9	31	2024-08-03 23:17:53	INV-202400000009	2064 Catherine Street	Port Michelleland	West Virginia	Dominican Republic	68873	256.07	Bank
10	35	2025-01-17 16:38:46	INV-202500000010	29201 Matthew Wells	Lisatown	Colorado	Reunion	39249	551.14	Buy N
11	9	2024-05-29 07:28:01	INV-202400000011	63352 Chapman Garden Suite 238	East Emily	New Jersey	Thailand	02395	236.59	Bank
12	18	2024-05-11 23:12:06	INV-202400000012	3156 Lewis Loaf	Adamchester	New Hampshire	Italy	64121	475.78	Gift C
13	36	2024-11-29 00:15:02	INV-202400000013	494 Smith Loop	Salinasburgh	Arizona	Suriname	06133	15.28	Credi
14	4	2025-08-13 02:19:55	INV-202500000014	020 Blackwell Village	West Williamfurt	California	Somalia	98995	363.56	Gift C
15	5	2024-10-13 09:17:39	INV-202400000015	1173 Kara Pines Apt. 594	New William	Wyoming	Paraguay	08260	18.43	Gift C
16	28	2024-10-21 07:04:05	INV-202400000016	1453 Victoria Branch Suite 222	Richardhaven	Vermont	Bermuda	59883	330.44	Gift C
17	47	2024-07-19 06:44:30	INV-202400000017	1414 Mark Inlet	Port Kylefort	Massachusetts	American Samoa	93529	171.86	Buy N
18	27	2024-04-20 00:05:08	INV-202400000018	5265 Michael Fork	Middletonchester	Washington	Tuvalu	65369	19.37	Buy N
19	43	2025-02-02 19:30:51	INV-202500000019	774 Diaz Camp	Amyburgh	Louisiana	Czech Republic	48208	32.58	Bank
20	33	2024-11-16 22:49:57	INV-202400000020	51448 Curtis Parkways	Frenchview	Colorado	Iceland	34378	594.34	Credi
21	12	2025-11-09 02:19:36	INV-202500000021	848 Medina Shoals Suite 726	Lake Coreyville	Louisiana	Panama	41751	745.24	Buy N
22	31	2025-07-15 00:06:32	INV-202500000022	2064 Catherine Street	Port Michelleland	West Virginia	Dominican Republic	68873	411.81	Gift C
23	49	2025-02-04 08:51:18	INV-202500000023	19438 Sandra Lake Suite 717	Wilsonchester	Delaware	Russian Federation	91764	333.16	Buy N
24	32	2024-03-18 22:33:26	INV-202400000024	5130 Howard Common	Gonzalezside	New Mexico	Pakistan	99226	46.54	Cash
25	1	2024-07-20 08:45:44	INV-202400000025	3496 Nathan Well Suite 482	West Joanna	Washington	Bermuda	70105	210.87	Gift C
26	12	2024-08-08 04:45:43	INV-202400000026	848 Medina Shoals Suite 726	Lake Coreyville	Louisiana	Panama	41751	331.52	Gift C
27	3	2025-07-28 22:01:45	INV-202500000027	65000 Miller Union Apt. 053	South Christytown	South Dakota	Liechtenstein	43199	251.74	Credi
28	28	2025-03-24 21:06:08	INV-202500000028	1453 Victoria Branch Suite 222	Richardhaven	Vermont	Bermuda	59883	448.35	Credi
29	11	2025-08-18 10:24:46	INV-202500000029	653 Hall Underpass	West Samantha	Alabama	Andorra	20336	308.04	Credi
30	45	2024-12-14 05:50:42	INV-202400000030	646 Zachary Forest	South Markshire	Florida	Korea	51419	61.48	Buy N
31	10	2025-08-25 20:25:39	INV-202500000031	180 Allen Park Apt. 671	Johnport	Nevada	Mongolia	16833	502.19	Buy N

Conclusion

The custom automation tool successfully generated a total of **2,100 records** across four related tables. The generated data strictly adheres to all specified constraints, including complex foreign key relationships between Transactions, Users, and Products.

By automating this process, I was able to produce a high-volume, production-like dataset that eliminates the need for manual entry and ensures consistent, error-free data for future testing cycles.

Additional Note: For scenarios requiring direct database injection via SQL scripts or compatibility with legacy import tools, the Python script can also be configured to output standard CSV files (without Excel formatting) by simply swapping the save_as_excel function with the standard csv.writer library.

Self – Assessment

Criteria	Outcomes	Grade	Self-Assessed Grade
1	Category Data (50)	20	20
2	Product Data (1000)	30	30
3	User Account Data (50)	20	20
4	Transaction Data (1000)	30	30
	Total	**100**	**100**