

download restaurant data from below mentioned link

Download Data Link ->

https://github.com/shashank-mishra219/Confluent-Kafka-Setup/blob/main/restaurant_orders.csv

Complete the given below task to finish this assignment.

1. Setup Confluent Kafka Account

The screenshot shows the Confluent Cloud dashboard. At the top, it says "Welcome back to Confluent Cloud!" with a "View environments" button. Below this is a "Your Confluent overview" section with a summary of existing resources:

Environments	Clusters	Topics	Partitions	ksqlDB	Connectors	Links
1	1	1	3	0	0	0

Below the summary, there's a section about payment information. It states "There's no payment information for your account" and shows a "Free trial" progress bar with "2 days left" and "Remaining free credit" of "\$400/\$400". A "Recommended" button is present. To the right, it says "Enter payment to avoid disruptions" and lists benefits: "You won't be charged until your free trial is over", "Avoid service interruptions when your free trial ends", and "Cancel or change services anytime". An "Update payment info" button is at the bottom.

At the bottom, there are three recommended actions:

- ksqlDB Get Started Webinar: From Batch to Real-time: Learn how to get started with stream data processing
- Confluent Terraform provider: Automate the management of your Confluent Cloud resources with ease
- Sink data to S3: Integrate with S3 using S3 sink connector, with easy setup and no operational

2. Create one kafka topic named as "restaurant-take-away-data" with 3 partitions.

The screenshot shows the Confluent Cloud "Topics" page. The left sidebar contains navigation links: Cluster Overview, Dashboard, Networking, API Keys, Cluster Settings, Stream Lineage, Stream Designer, Topics (selected), ksqlDB, Connectors, Clients, and Schema Registry. The main content area has a "Topics" heading and a search bar. Below the search bar is a table listing topics:

Topic name	Partitions	Production (last min)	Consumption (last min)	Schema
restaurant-take-away-data	3	--	--	Set a schema
testtopic	3	--	--	Edit schema

At the top right of the table, there is a "+ Add topic" button. At the bottom of the page, there are links for "CLI and Tools" and "Support".

3. Setup key (string) & value (json) schema in the confluent schema registry.

The screenshot shows the Confluent Schema Registry interface for a topic named "restaurant-take-away-data". The "Schema" tab is selected, showing a JSON schema for the "Value" field. The schema is a JSON object with the following structure:

```
1 {
2   "id": "http://example.com/myURI.schema.json",
3   "schema": "http://json-schema.org/draft-07/schema#",
4   "additionalProperties": false,
5   "description": "Sample schema to help you get started.",
6   "properties": {
7     "Item Name": {
8       "description": "The type(v) type is used.",
9       "type": "string"
10    },
11    "Order Date": {
12      "description": "The type(v) type is used.",
13      "type": "string"
14    },
15    "Order Number": {
16      "description": "The type(v) type is used.",
17      "type": "number"
18    },
19    "Product Price": {
20      "description": "The type(v) type is used.",
21      "type": "number"
22    }
23  }
24 }
```

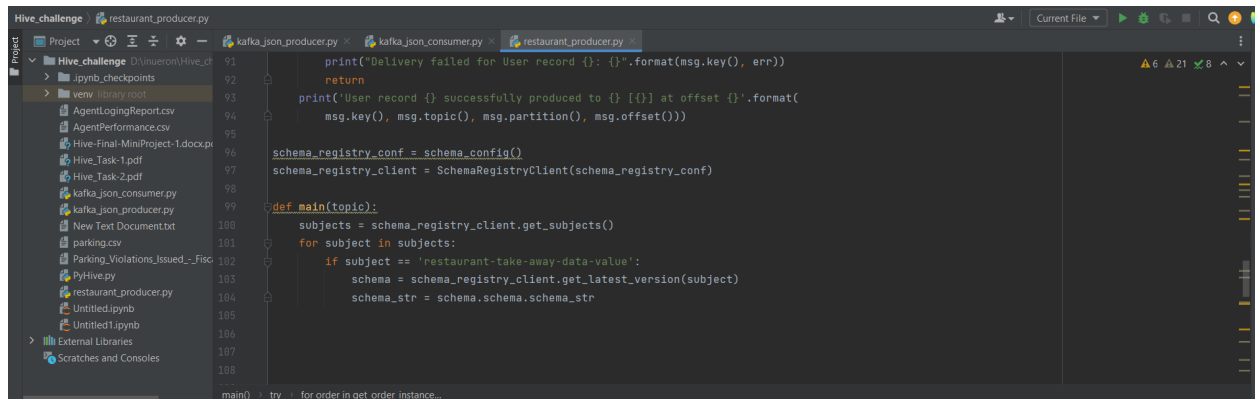
The right sidebar contains the following metadata:

- Description:** Add description
- Tags:** Add tags to this version
- Add business metadata:** Add business metadata
- Schema doc:** Sample schema to help you get started.
- Date created:** Oct. 28 2022 7:56 PM

The screenshot shows the Confluent Schema Registry interface for a topic named "restaurant-take-away-data". The "Value" tab is selected, showing a large empty text area for the "Value" field. The "Key" field is highlighted in blue. The interface is mostly empty, with a cursor visible in the text area.

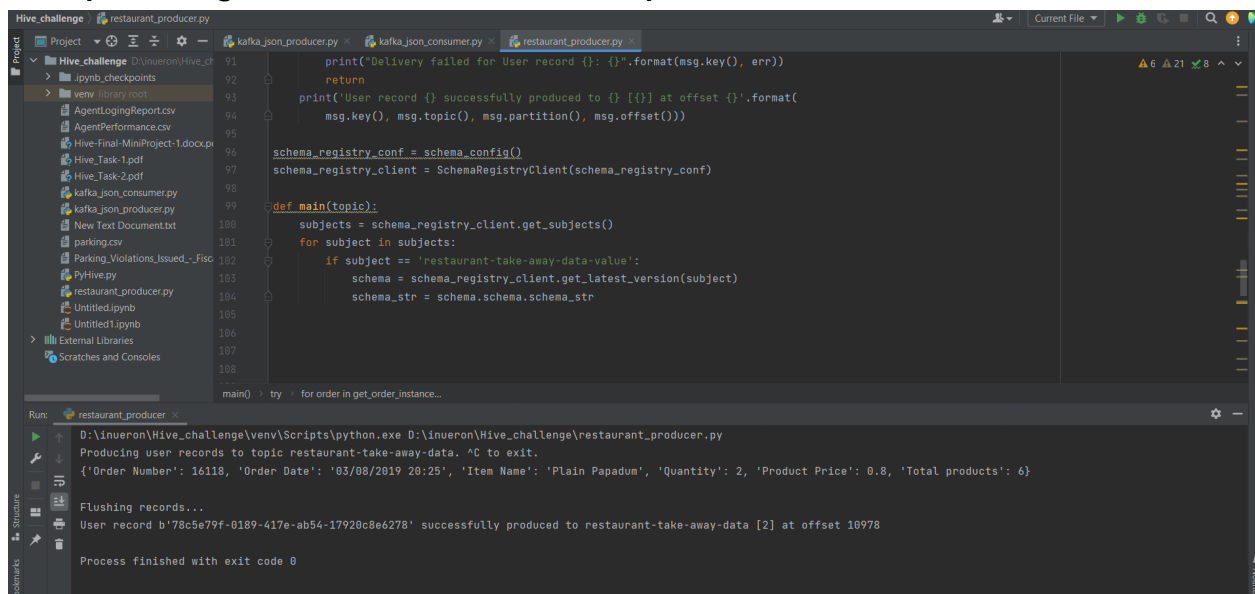
4. Write a kafka producer program (python or any other language) to read data records from restaurant data csv file,

make sure schema is not hardcoded in the producer code, read the latest version of schema and schema_str from schema registry and use it for data serialization.



```
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
```

5. From producer code, publish data in Kafka Topic one by one and use dynamic key while publishing the records into the Kafka Topic



```
116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
```

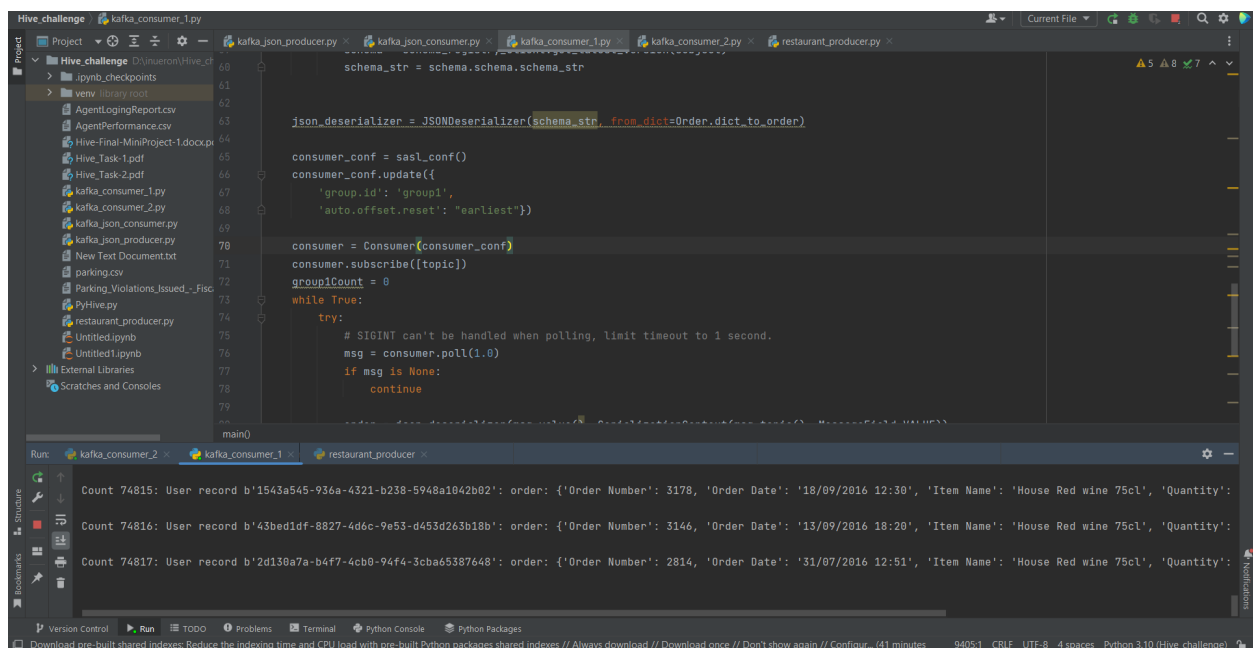
6. Write kafka consumer code and create two copies of same consumer code and save it with different names (kafka_consumer_1.py & kafka_consumer_2.py), again make sure latest schema version and schema_str is not hardcoded in the consumer code, read it automatically from the schema registry to deserialize the data.

Now test two scenarios with your consumer code:

a.) Use "group.id" property in consumer config for both consumers and mention different group_ids in kafka_consumer_1.py & kafka_consumer_2.py, apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer consumed and printed on the terminal

Kafka_consumer_1

Total records: 74817

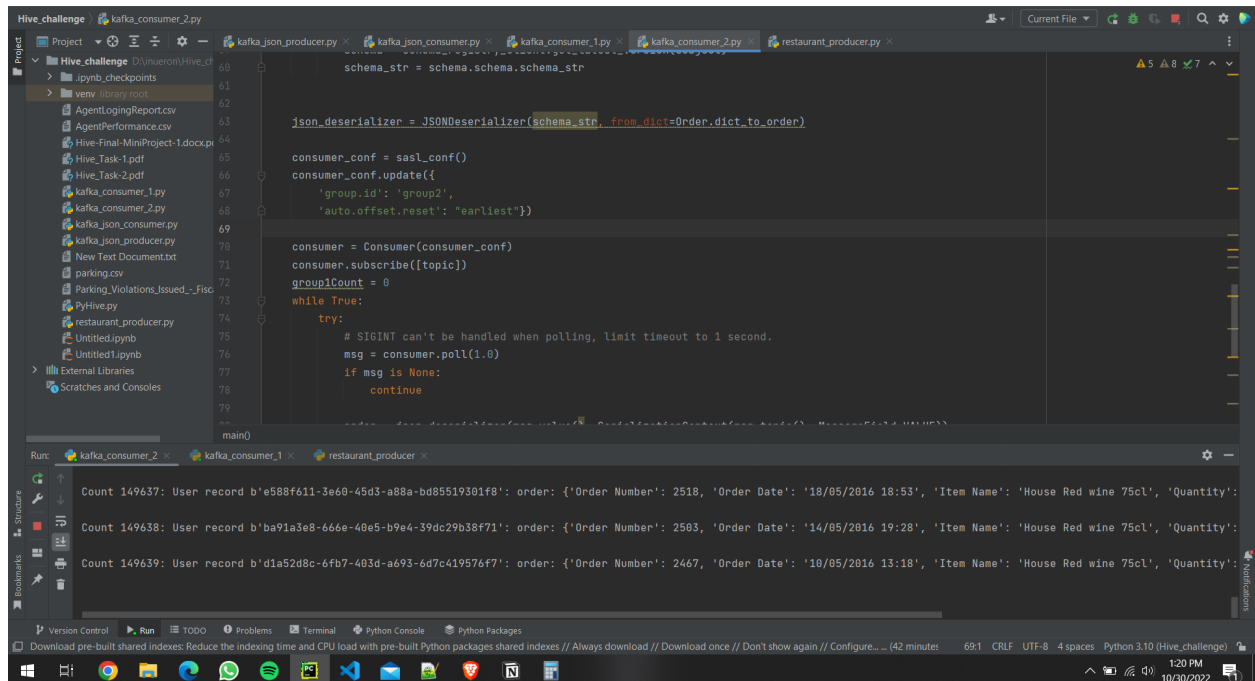


The screenshot shows an IDE with the following components:

- Project Explorer:** Lists files in the 'Hive_challenge' project, including 'kafka_consumer_1.py'.
- Code Editor:** Displays the code for 'kafka_consumer_1.py'. The code includes:
 - Schema loading: `schema_str = schema.schema.schema_str`
 - JSON deserializer: `json_deserializer = JSONDeserializer(schema_str, from_dict=Order.dict_to_order)`
 - Consumer configuration: `consumer_conf = sasl_conf()` and `consumer_conf.update({'group.id': 'group1', 'auto.offset.reset': 'earliest'})`
 - Consumer creation and subscription: `consumer = Consumer(consumer_conf)` and `consumer.subscribe([topic])`
 - Counting and polling: `group1Count = 0` and a `while True:` loop with `msg = consumer.poll(1.0)`
 - Message processing: A `try:` block that prints the message and increments `group1Count`.
- Run Console:** Shows the output of the program, displaying three records with their IDs and details:
 - Count 74815: User record b'1543a545-936a-4321-b238-5948a1042b02': order: {'Order Number': 3178, 'Order Date': '18/09/2016 12:30', 'Item Name': 'House Red wine 75cl', 'Quantity': ...}
 - Count 74816: User record b'43bed1df-8827-4d6c-9e53-d453d263b18b': order: {'Order Number': 3146, 'Order Date': '13/09/2016 18:20', 'Item Name': 'House Red wine 75cl', 'Quantity': ...}
 - Count 74817: User record b'2d139a7a-b4f7-4cb0-94f4-3cba65387648': order: {'Order Number': 2814, 'Order Date': '31/07/2016 12:51', 'Item Name': 'House Red wine 75cl', 'Quantity': ...}

Kafka_consumer_2

Total records: 149639



```
60 schema_str = schema.schema.schema_str
61
62 json_deserializer = JSONDeserialzer(schema_str, from_dict=Order.dict_to_order)
63
64 consumer_conf = sasl_conf()
65 consumer_conf.update({
66     'group.id': 'group2',
67     'auto.offset.reset': 'earliest'})
68
69 consumer = Consumer(consumer_conf)
70 consumer.subscribe([topic])
71 group1Count = 0
72 while True:
73     try:
74         # SIGINT can't be handled when polling, limit timeout to 1 second.
75         msg = consumer.poll(1.0)
76         if msg is None:
77             continue
78
79         # ... process message ...
80
81     except KeyboardInterrupt:
82         break
83
84 main()
```

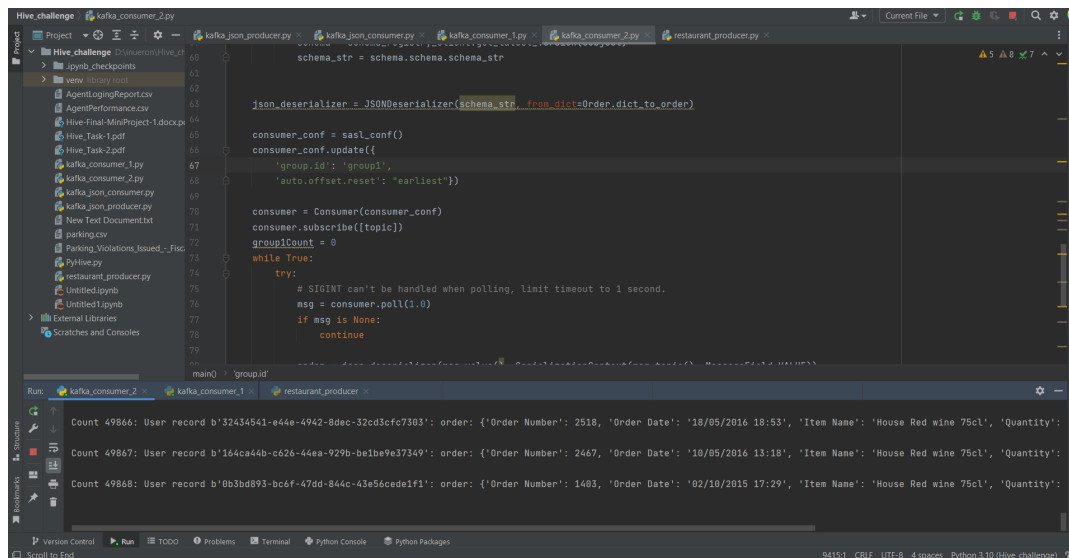
Run: kafka_consumer_2 kafka_consumer_1 restaurant_producer

```
Count 149637: User record b'e588f611-3e60-45d3-a88a-bd85519301f8': order: {'Order Number': 2518, 'Order Date': '18/05/2016 18:53', 'Item Name': 'House Red wine 75cl', 'Quantity':
Count 149638: User record b'ba91a3e8-666e-40e5-b9e4-39dc29b38f71': order: {'Order Number': 2503, 'Order Date': '14/05/2016 19:28', 'Item Name': 'House Red wine 75cl', 'Quantity':
Count 149639: User record b'd1a52d8c-6fb7-403d-a693-6d7c419576f7': order: {'Order Number': 2467, 'Order Date': '10/05/2016 13:18', 'Item Name': 'House Red wine 75cl', 'Quantity':
```

b.) Use "group.id" property in consumer config for both consumers and mention same group_ids in kafka_consumer_1.py & kafka_consumer_2.py, apply "earliest" offset property in both consumers and run these two consumers from two different terminals. Calculate how many records each consumer consumed and printed on the terminal

Kafka_consumer_2

Total records - 49868



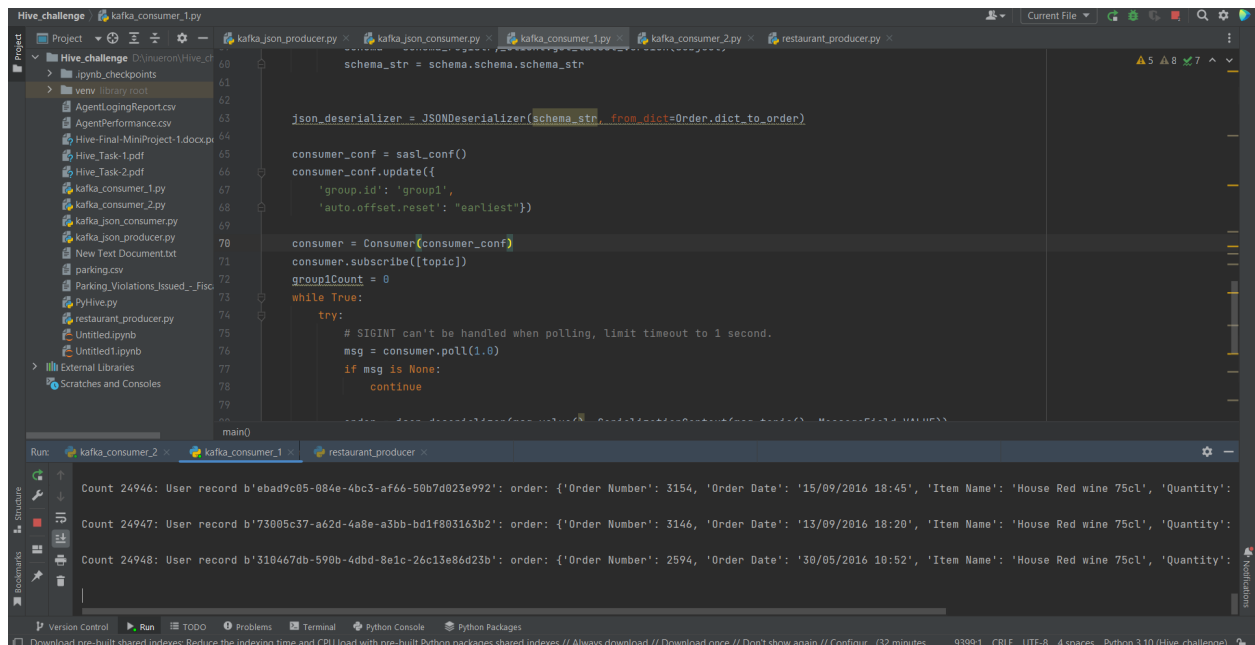
```
60 schema_str = schema.schema.schema_str
61
62 json_deserializer = JSONDeserialzer(schema_str, from_dict=Order.dict_to_order)
63
64 consumer_conf = sasl_conf()
65 consumer_conf.update({
66     'group.id': 'group1',
67     'auto.offset.reset': 'earliest'})
68
69 consumer = Consumer(consumer_conf)
70 consumer.subscribe([topic])
71 group1Count = 0
72 while True:
73     try:
74         # SIGINT can't be handled when polling, limit timeout to 1 second.
75         msg = consumer.poll(1.0)
76         if msg is None:
77             continue
78
79         # ... process message ...
80
81     except KeyboardInterrupt:
82         break
83
84 main()
```

Run: kafka_consumer_2 kafka_consumer_1 restaurant_producer

```
Count 49866: User record b'32434541-e44e-4942-8dec-32cd3cfc7303': order: {'Order Number': 2518, 'Order Date': '18/05/2016 18:53', 'Item Name': 'House Red wine 75cl', 'Quantity':
Count 49867: User record b'164ca44b-c626-44ea-929b-be1be9e37349': order: {'Order Number': 2467, 'Order Date': '10/05/2016 13:18', 'Item Name': 'House Red wine 75cl', 'Quantity':
Count 49868: User record b'0b3bd893-bc6f-47dd-844c-43e56cede1f1': order: {'Order Number': 1403, 'Order Date': '02/10/2015 17:29', 'Item Name': 'House Red wine 75cl', 'Quantity':
```

Kafka_consumer_1

Total records - 24948



The screenshot shows a code editor with a Python script for a Kafka consumer. The script defines a schema, a JSON deserializer, a consumer configuration, and a consumer object. It subscribes to a topic and polls for messages. The output window shows three sample records being processed.

```
schema_str = schema.schema.schema_str

json_deserializer = JSONDeserializer(schema_str, from_dict=Order.dict_to_order)

consumer_conf = sasl_conf()
consumer_conf.update({
    'group.id': 'group1',
    'auto.offset.reset': 'earliest'})

consumer = Consumer(consumer_conf)
consumer.subscribe([topic])
group1Count = 0
while True:
    try:
        # SIGINT can't be handled when polling, limit timeout to 1 second.
        msg = consumer.poll(1.0)
        if msg is None:
            continue
        order = json_deserializer.from_dict(msg.value)
        print(f'Count {group1Count}: User record {msg.key}: {order}')
        group1Count += 1
    except KeyboardInterrupt:
        break
    except Exception as e:
        print(f'Exception: {e}')
        continue
    except:
        continue
main()
```

Count 24946: User record b'ebad9c05-084e-4bc3-af66-50b7d023e992': order: {'Order Number': 3154, 'Order Date': '15/09/2016 18:45', 'Item Name': 'House Red wine 75cl', 'Quantity': 1}

Count 24947: User record b'73085c37-a62d-4a8e-a3bb-bd1f803163b2': order: {'Order Number': 3146, 'Order Date': '13/09/2016 18:20', 'Item Name': 'House Red wine 75cl', 'Quantity': 1}

Count 24948: User record b'310467db-590b-4dbd-8e1c-26c13e86d23b': order: {'Order Number': 2594, 'Order Date': '30/05/2016 10:52', 'Item Name': 'House Red wine 75cl', 'Quantity': 1}

7. Once above questions are done, write another kafka consumer to read data from kafka topic and from the consumer code create one csv file "output.csv" and append consumed records output.csv file

Solution attached in githu