به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

مقدمه ای بر علوم داده

تمرین کامپیوتری شماره۲

نام و نام خانوادگی	محمد عسکری
شماره دانشجویی	۸۱۰۱۹۸۴۴۱
تاریخ ارسال گزارش	۱۴۰۴/۰۲/۱۱

Real-Time Payment Data Pipeline

Introduction

This report documents the design and implementation of a real-time data pipeline for processing financial transactions, developed as part of the "Introduction to Data Science" course at the University of Tehran. The pipeline simulates transaction flows from a fictional payment provider named "Darooghe," aiming to handle large-scale streaming data, detect anomalies, and provide actionable insights.

The system integrates Apache Kafka for data ingestion, Apache Spark for both batch and stream processing, MongoDB and PostgreSQL for storage, and Python-based tools for validation and visualization. Each component was configured and tested in a Linux-based environment to emulate real-world conditions.

---

1. Environment Setup and Data Generation

1.1 Technology Stack

We utilized the following core technologies:

- **Java (OpenJDK 11):** Required for running Kafka and Spark.
- **Apache Kafka:** Acts as the real-time message broker.
- **PySpark:** Python API for Apache Spark, used for distributed processing.
- **Confluent Kafka (Python client):** Enables Kafka interaction from Python.
- **Virtual Environment:** Ensures isolated and reproducible Python dependencies.

1.2 Installation and Configuration

A virtual machine (Linux environment via WSL2) was set up. Java was installed using OpenJDK 11. Kafka was configured using standard scripts to start both the ZooKeeper and Kafka broker services. Spark was installed alongside PySpark.

Kafka topics were created manually and accessed using both CLI and Python scripts. A virtual Python environment was activated to manage the Python packages, including `confluent_kafka`, `pyspark`, `pymongo`, and `psycopg2`.

1.3 Transaction Generator

(`darooghe_pulse.py`) generates both historical (backfilled) and live events using a Non-Homogeneous Poisson Process.

Key parameters were adjustable through environment variables, including the event rate, peak-hour multiplier, and fraud rate. This script produced 20,000 historical events and then continuously streamed transactions to the Kafka topic `darooghe.transactions`.

Kafka CLI tools were used to validate that data was properly written to the topic.

```
{"transaction_id": "ec40c1e7-6990-4330-843b-951f63cb2ca2", "timestamp": "202
5-04-15T08:16:49.038665Z", "customer_id": "cust_4463", "merchant_id": "merch
_45", "merchant_category": "food_service", "payment_method": "mobile", "amou
nt": 649311, "location": {"lat": 35.70630654600936, "lng": 51.3705068017449}
, "device_info": {"os": "iOS", "app_version": "3.1.0", "device_model": "iPho
ne 15"}, "status": "approved", "commission_type": "tiered", "commission_amou
nt": 12986, "vat_amount": 58437, "total_amount": 707748, "customer_type": "i
ndividual", "risk_level": 2, "failure_reason": null}
{"transaction_id": "738dc608-f7b3-4874-afc7-496a659705be", "timestamp": "202
5-04-15T08:16:51.910443Z", "customer_id": "cust_250", "merchant_id": "merch_
66", "merchant_category": "retail", "payment_method": "online", "amount": 73
5754, "location": {"lat": 35.69310489684208, "lng": 51.339912555841885}, "de
vice_info": {"os": "Android", "app_version": "1.9.5", "device_model": "Googl
e Pixel 6"}, "status": "approved", "commission_type": "tiered", "commission_
amount": 14715, "vat_amount": 66217, "total_amount": 801971, "customer_type"
: "individual", "risk_level": 1, "failure_reason": null}
{"transaction_id": "26eaa58d-73ed-4fa2-aa05-a5a9cf2db775", "timestamp": "202
5-04-15T08:16:52.137493Z", "customer_id": "cust_4126", "merchant_id": "merch
_29", "merchant_category": "government", "payment_method": "pos", "amount":
1856686, "location": {"lat": 35.681901728793825, "lng": 51.31688019109848},
"device_info": {}, "status": "declined", "commission_type": "flat", "commiss
ion_amount": 37133, "vat_amount": 167101, "total_amount": 2023787, "customer
_type": "CIP", "risk_level": 1, "failure_reason": "fraud_prevented"}
{"transaction_id": "9548c4ca-7419-4f32-8b09-59285b38aa35", "timestamp": "202
5-04-15T08:16:52.567015Z", "customer_id": "cust_2070", "merchant_id": "merch
_81", "merchant_category": "entertainment", "payment_method": "online", "amo
unt": 1741153, "location": {"lat": 35.7247898183667, "lng": 51.3686399820027
}, "device_info": {"os": "Android", "app_version": "2.4.1", "device_model":
"Samsung Galaxy S25"}, "status": "declined", "commission_type": "tiered", "c
ommission_amount": 34823, "vat_amount": 156703, "total_amount": 1897856, "cu
stomer_type": "individual", "risk_level": 1, "failure_reason": "insufficient
_funds"}
{"transaction_id": "a4db3c46-710c-4c26-b5ba-8564a2069a2a", "timestamp": "202
5-04-15T08:16:52.593358Z", "customer_id": "cust_1007", "merchant_id": "merch
_94", "merchant_category": "food_service", "payment_method": "nfc", "amount"
: 1631786, "location": {"lat": 35.70384425004735, "lng": 51.32648660188084},
 "device_info": {}, "status": "approved", "commission_type": "progressive",
"commission_amount": 32635, "vat_amount": 146860, "total_amount": 1778646, "
customer_type": "individual", "risk_level": 3, "failure_reason": null}
^CProcessed a total of 20297 messages
m4skari@LAPTOP-O6VNC6EO:~/kafka_2.13-3.5.1$
```


Sample output of real-time transaction generator published to `darooghe.transactions` Kafka topic.

---

2. Data Ingestion and Validation

2.1 Kafka Consumer and Validator

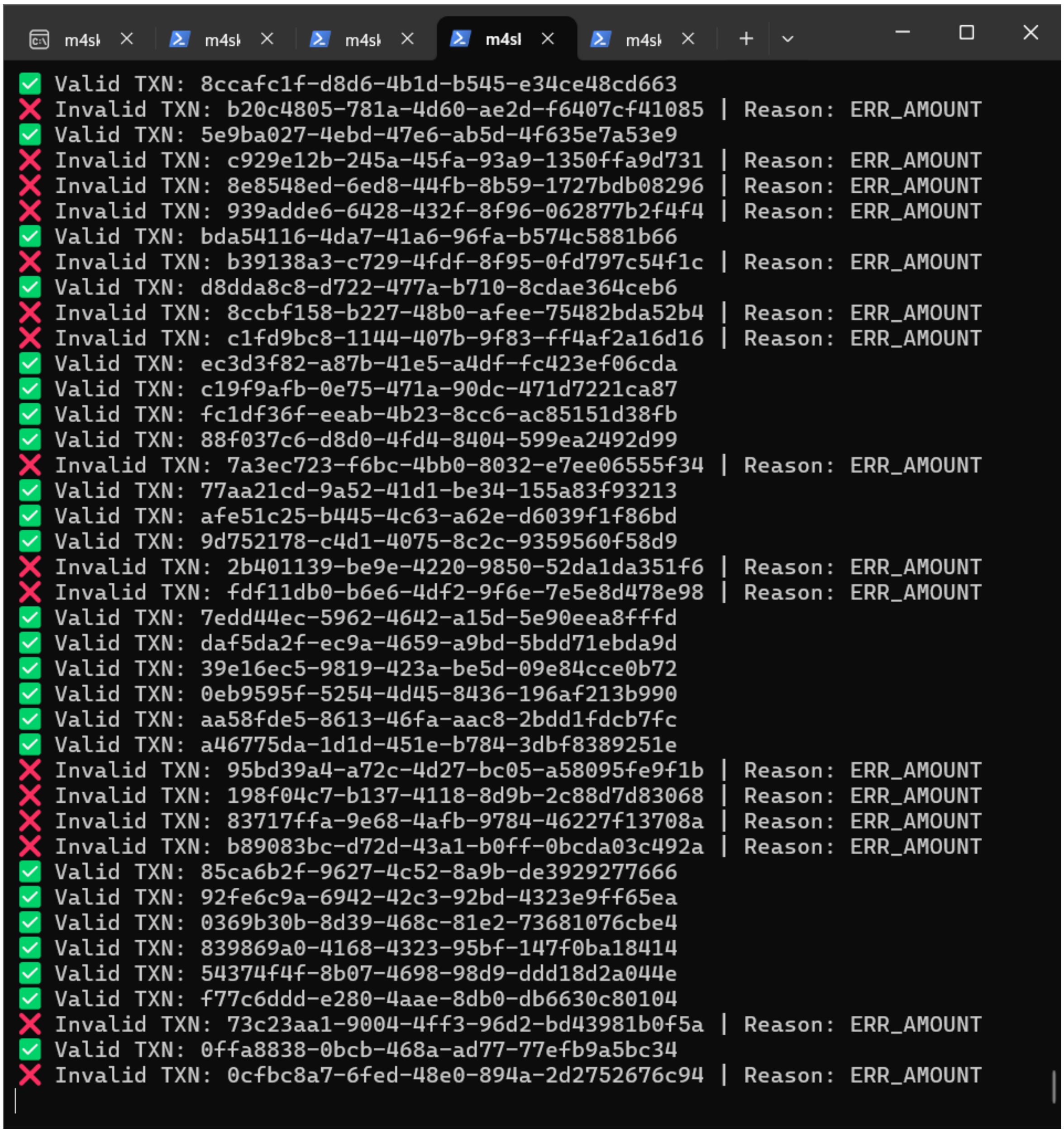
The `transaction_validator.py` script acts as a Kafka consumer, reading from the `darooghe.transactions` topic and applying a set of validation rules before forwarding invalid records to the `darooghe.error_logs` topic.

2.2 Validation Logic

Each transaction was checked against three main rules:

- **Amount Consistency:** Ensures `total_amount = amount + vat_amount + commission_amount`. Violations trigger `ERR_AMOUNT`.
- **Time Warping:** Filters out transactions from the future or older than one day (`ERR_TIME`).
- **Device Mismatch:** Checks that mobile transactions originate from iOS or Android devices (`ERR_DEVICE`).

Invalid records were serialized into JSON format and sent to the error topic. Console outputs and Kafka CLI confirmed correct logging behavior.



Real-time output of Kafka consumer showing valid and invalid transactions with associated error codes.

---

3. Batch Processing and Analysis

3.1 Objective

Batch analysis focused on identifying patterns in historical data, such as temporal trends and customer segmentation.

3.2 Dataset and Tools

The dataset was exported from Kafka into a JSON file and loaded into PySpark. Tools used include PySpark DataFrame API, SQL functions, and timestamp transformation utilities.

3.3 Temporal Analysis

Using Spark SQL, transaction frequency was aggregated by hour of day and day of week. Results showed:

- Peak activity around noon and 19 PM.
- Highest transaction volume on Tuesday.

```
=== Most Active Hours ===
+-----------+-----+
|hour_of_day|count|
+-----------+-----+
|0          |830  |
|1          |816  |
|2          |849  |
|3          |824  |
|4          |825  |
|5          |879  |
|6          |828  |
|7          |898  |
|8          |819  |
|9          |823  |
|10         |878  |
|11         |834  |
|12         |7214 |
|13         |817  |
|14         |816  |
|15         |797  |
|16         |818  |
|17         |852  |
|18         |821  |
|19         |7061 |
|20         |4142 |
|21         |877  |
|22         |801  |
|23         |804  |
+-----------+-----+
```

```
+-----------+-----+

=== Most Active Days (Sunday=1) ===
+-----------+-----+
|day_of_week|count|
+-----------+-----+
|1          |2960 |
|2          |2863 |
|3          |18783|
|4          |2837 |
|5          |2858 |
|6          |2834 |
|7          |2788 |
+-----------+-----+
```

3.4 Customer Segmentation

Customers were grouped based on their transaction frequency:

- **Frequent Users:** ≥ 10 transactions.
- **Normal Users:** 3–9 transactions.
- **Rare Users:** < 3 transactions.

The segmentation logic was implemented using the `when()` and `otherwise()` functions in PySpark. This information can support downstream marketing or anomaly detection.

```
=== Customer Segmentation by Activity ===
+-----------+---------+--------+
|customer_id|txn_count|segment |
+-----------+---------+--------+
|cust_1314  |18       |Frequent|
|cust_3234  |17       |Frequent|
|cust_1979  |17       |Frequent|
|cust_352   |17       |Frequent|
|cust_3890  |17       |Frequent|
|cust_1476  |16       |Frequent|
|cust_3624  |16       |Frequent|
|cust_3614  |16       |Frequent|
|cust_1618  |16       |Frequent|
|cust_905   |16       |Frequent|
|cust_2384  |16       |Frequent|
|cust_2634  |16       |Frequent|
|cust_2673  |16       |Frequent|
|cust_2779  |15       |Frequent|
|cust_1832  |15       |Frequent|
|cust_2616  |15       |Frequent|
|cust_2434  |15       |Frequent|
|cust_3882  |15       |Frequent|
|cust_1495  |15       |Frequent|
|cust_1161  |15       |Frequent|
+-----------+---------+--------+
only showing top 20 rows
```

---

4. Real-Time Streaming and PostgreSQL Integration

4.1 Spark Structured Streaming

We built a real-time Spark streaming job (`realtime_processor.py`) to process Kafka data and apply simple fraud detection rules. The rules included:

- Late-night transactions (00:00–06:00).
- High-risk transactions with large amounts.
- Device anomalies in mobile transactions.

```
{"transaction_id":"39c501a2-490c-4ef2-8743-210e73006a1d","timestamp":"2025-05-01T05:19:53.794275Z","amount":1766139,"risk_level":3,"payment_method":"mobile","device_info":{"os":"iOS"},"ts":"2025-05-01T08:49:53.794+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"9543f208-bf4a-42b1-932b-38bc9921cf51","timestamp":"2025-05-01T05:20:01.941585Z","amount":1227489,"risk_level":3,"payment_method":"nfc","device_info":{},"ts":"2025-05-01T08:50:01.941+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"5f13d5a3-3cf4-4373-b7aa-22615f44c5ce","timestamp":"2025-05-01T05:20:03.059338Z","amount":1629111,"risk_level":3,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:03.059+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"66119c10-7cd0-481c-a9be-9e9ce3424a7e","timestamp":"2025-05-01T05:20:03.365055Z","amount":1629404,"risk_level":3,"payment_method":"nfc","device_info":{},"ts":"2025-05-01T08:50:03.365+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"32079855-cc16-447d-9fac-c6d335cc01ce","timestamp":"2025-05-01T05:20:05.557157Z","amount":1961618,"risk_level":3,"payment_method":"mobile","device_info":{"os":"iOS"},"ts":"2025-05-01T08:50:05.557+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"9d090a50-16a1-4aa5-abb6-b4f698828c5c","timestamp":"2025-05-01T05:20:05.762966Z","amount":1766187,"risk_level":5,"payment_method":"pos","device_info":{},"ts":"2025-05-01T08:50:05.762+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"77814fa6-5d8b-4bd2-af8d-c83c871f3ad7","timestamp":"2025-05-01T05:20:07.423564Z","amount":1706527,"risk_level":5,"payment_method":"nfc","device_info":{},"ts":"2025-05-01T08:50:07.423+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"cb709ff6-2221-4c1c-bf0b-520bd3c5038d","timestamp":"2025-05-01T05:20:09.040569Z","amount":1140195,"risk_level":5,"payment_method":"online","device_info":{"os":"iOS"},"ts":"2025-05-01T08:50:09.040+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"ce25476c-e233-4df9-aaec-fea3bd194973","timestamp":"2025-05-01T05:20:18.708228Z","amount":1565725,"risk_level":3,"payment_method":"pos","device_info":{},"ts":"2025-05-01T08:50:18.708+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"308eaf07-e322-40a0-9792-a6a3496cc1ab","timestamp":"2025-05-01T05:20:20.380822Z","amount":1851236,"risk_level":5,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:20.380+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"918402f9-b015-4559-9ef8-20ad6839d3a4","timestamp":"2025-05-01T05:20:20.400054Z","amount":1516269,"risk_level":3,"payment_method":"pos","device_info":{},"ts":"2025-05-01T08:50:20.400+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"48a641ae-b689-4c7d-bcf1-72f2a3d7ac75","timestamp":"2025-05-01T05:20:23.796009Z","amount":1671465,"risk_level":3,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:23.796+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"e53ac748-614d-4609-b577-62a8ea6b6225","timestamp":"2025-05-01T05:20:26.790301Z","amount":1702329,"risk_level":3,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:26.790+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"da5351fb-07fb-48c6-974b-e0b84ae418cc","timestamp":"2025-05-01T05:20:37.482187Z","amount":1667443,"risk_level":3,"payment_method":"pos","device_info":{},"ts":"2025-05-01T08:50:37.482+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"f1a7b893-9544-4594-9099-2e1e1f97009f","timestamp":"2025-05-01T05:20:40.285357Z","amount":1299379,"risk_level":3,"payment_method":"mobile","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:40.285+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"10d16b12-4928-415b-b647-908a1919556a","timestamp":"2025-05-01T05:20:40.653266Z","amount":1919221,"risk_level":3,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:40.653+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"58812201-779b-4010-ad21-c95fa04249d2","timestamp":"2025-05-01T05:20:41.505867Z","amount":1633441,"risk_level":3,"payment_method":"nfc","device_info":{},"ts":"2025-05-01T08:50:41.505+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"283b0580-003b-4a67-93f8-4744f84c9e6b","timestamp":"2025-05-01T05:20:41.742277Z","amount":1499679,"risk_level":3,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:41.742+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"f2869e3c-31ea-4bc9-9ce4-b7fa5248b09e","timestamp":"2025-05-01T05:20:43.189965Z","amount":1892338,"risk_level":3,"payment_method":"online","device_info":{"os":"Android"},"ts":"2025-05-01T08:50:43.189+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
{"transaction_id":"30389c04-d035-44eb-aba8-60cd71c8438e","timestamp":"2025-05-01T05:20:43.989519Z","amount":1720801,"risk_level":3,"payment_method":"nfc","device_info":{},"ts":"2025-05-01T08:50:43.989+03:30","hour":8,"alert_reason":"HIGH_RISK_HIGH_AMOUNT"}
```


4.2 Database Sink

Detected fraud alerts were sent to a PostgreSQL database using the `psycopg2` Python client. The Spark job used `foreachBatch()` to write each micro-batch.

A checkpointing mechanism was implemented to maintain fault tolerance. The final schema included transaction ID, timestamp, risk level, and alert reason.

SQL queries confirmed correct insertion into the database. Duplicate records were handled using `ON CONFLICT DO NOTHING`.

---

5. Data Visualization

5.1 Objective

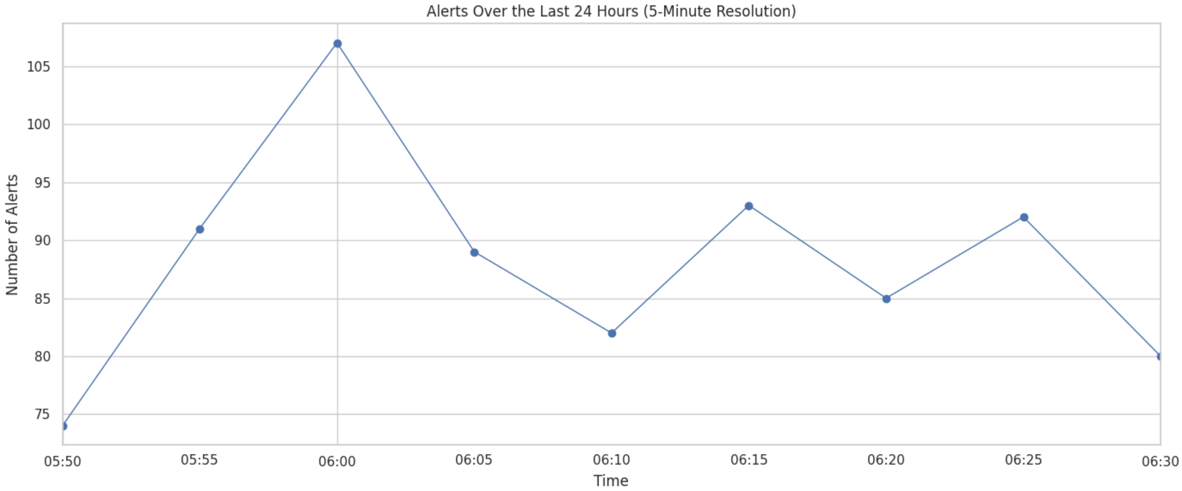
Visualizations were created to analyze trends in detected fraud events stored in the PostgreSQL database.

5.2 Tools

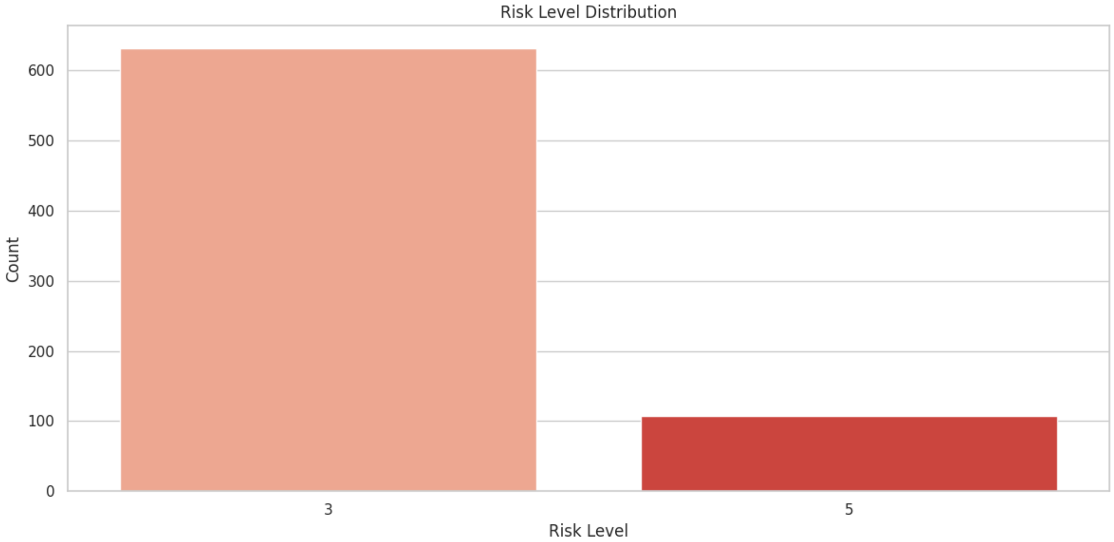
Python packages used include `psycopg2`, `pandas`, `matplotlib`, and `seaborn`. Data was loaded from PostgreSQL using SQL queries.

5.3 Charts and Insights

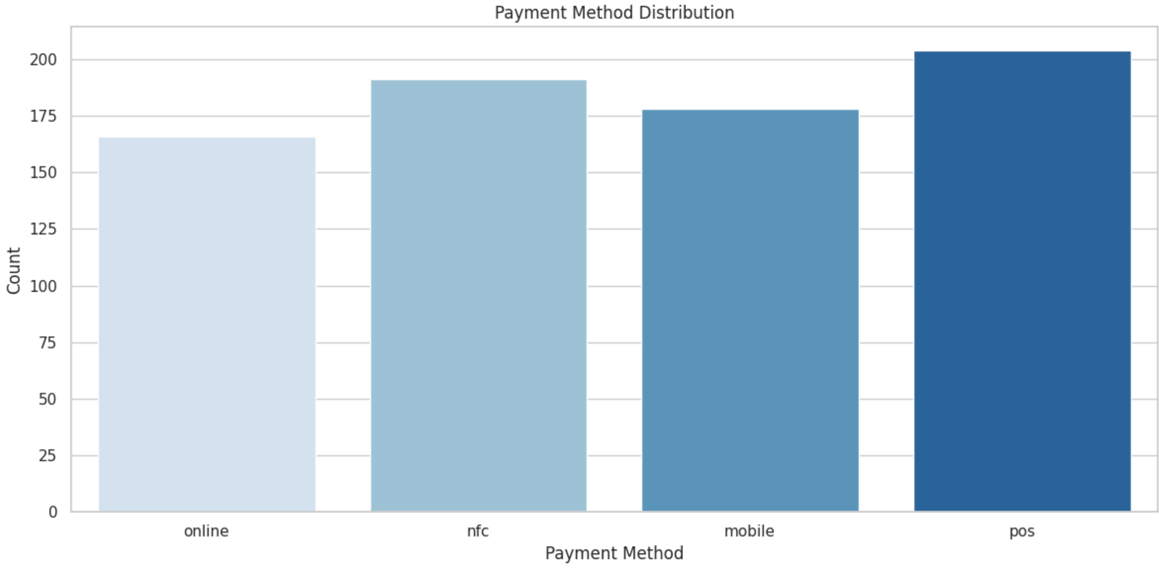
- **Alerts Over Time:** Time-series plot showed real-time fraud activity.



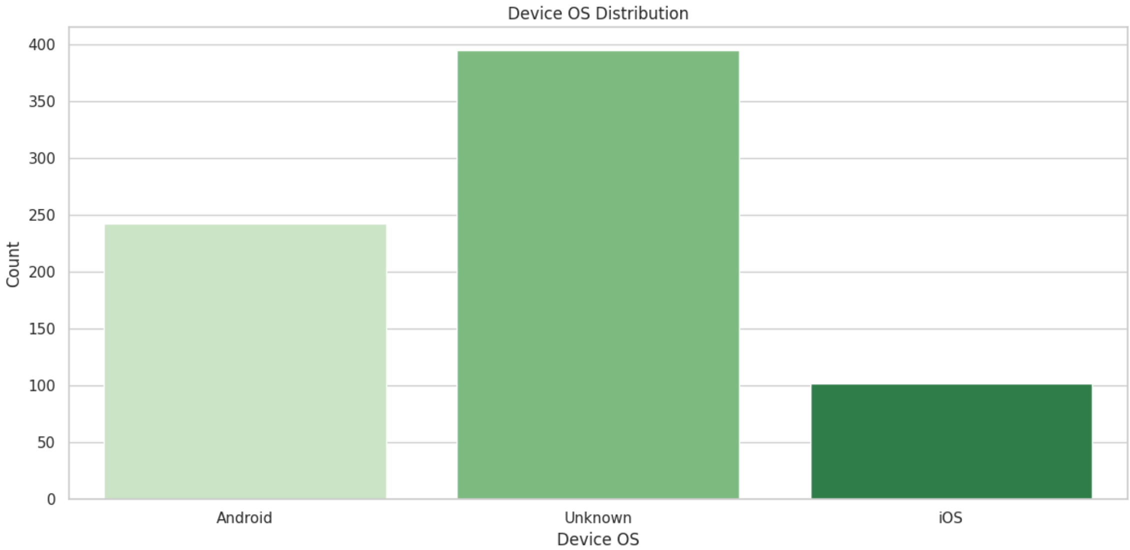
- **Risk Level Distribution:** High concentration in risk level 3.



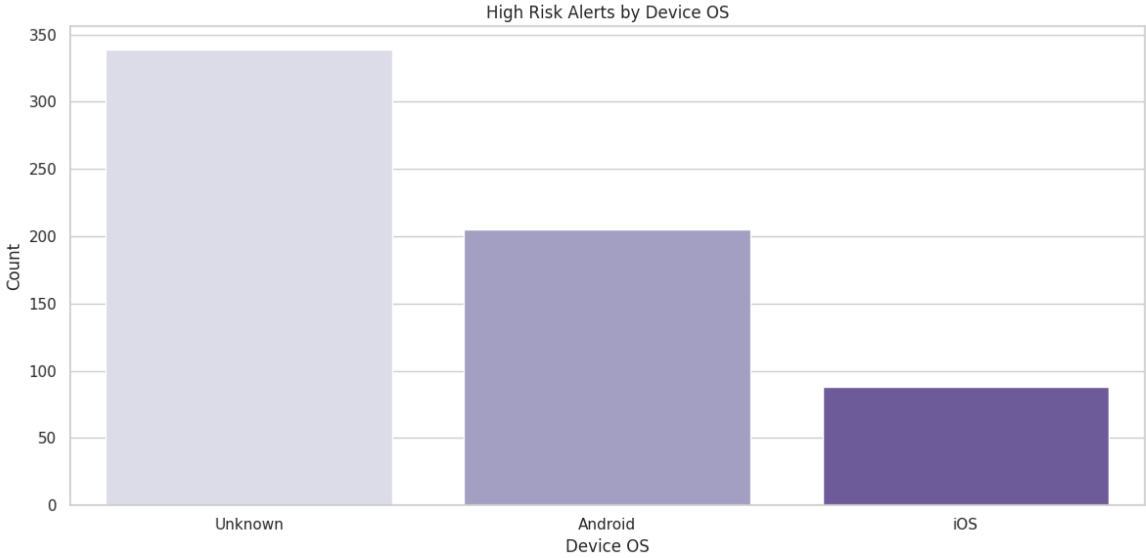
- **Payment Method Distribution:** Fraudulent activity was spread across POS, NFC, and Mobile.



- **Device OS Breakdown:** Android was the most common among flagged mobile transactions.



- **High-Risk Devices:** Certain OS types were repeatedly linked to high-risk alerts.



Conclusion

This project successfully demonstrated the architecture and implementation of a real-time data pipeline for transaction processing. By leveraging Kafka, Spark, and PostgreSQL, the system supports continuous ingestion, validation, fraud detection, and visualization.

Each component was modular and extensible, making the pipeline suitable for scaling in production environments. Potential improvements include advanced fraud detection rules, dynamic pricing strategies, and real-time dashboards.

The pipeline met its functional goals and proved to be a reliable platform for streaming financial data analytics.