

159-实习内容准备

实际开发前准备

实际开发内容

要求1

要求2

要求3

要求4

要求5

要求6

要求7

实际开发前准备

第一周基本上：领电脑，配置公司开发环境，申请权限，看文档，没有特别多的开发任务。

具体需要本地安装的必备软件：

开发工具：JDK、IDEA、Cursor、Sublime、PyCharm、Xshell

数据库管理工具：DataGrip、RDM、

本地测试工具：Postman、Jmeter

容器管理工具：Docker、rancher

依赖管理：maven

git管理工具：SourceTree

其他：kafkatool

申请权限：

主要有：内部知识库 Confluence 、Git代码仓库、JIRA、Jenkins、Grafana、Prometheus、ELK、Nexus 、Nacos

MySQL、Redis测试数据库地址

实际开发内容

金锐这里做一个完整一点有挑战性的需求，涵盖各种核心技术和业务。

要求1

实现一个接口，能够随机生成二代征信报文，具体的报文格式参考下面的JSON

```
1  {
2    // ===== 报告头信息 ===== //
3    "report_header": {
4      // 报告生成时间 (ISO8601标准格式, 精确到秒)
5      "report_query_time": "2025-07-19T14:30:25+08:00",
6
7      // 是否查询到有效征信记录 (true:有信贷历史, false:可能为白户)
8      "has_valid_credit_info": true,
9
10     // 首次贷款/贷记卡距今月份数 (反映信用历史长度, ≥60月为优质客户)
11     "first_credit_month_diff": 68
12   },
13
14   // ===== 身份信息 ===== //
15   "identity_info": {
16     // 姓名 (需与身份证完全一致)
17     "name": "张三",
18
19     // 其他证件列表 (非身份证证件补充)
20     "other_certificates": [
21       {
22         // 证件类型 (1:身份证(默认) 2:护照 5:港澳通行证 8:外国人居留证)
23         "cert_type": "2",
24         // 证件号码 (需脱敏展示)
25         "cert_number": "E12345678"
26       }
27     ],
28
29     // 婚姻状况 (10:未婚 20:已婚 40:离异 91:单身)
30     "marital_status": "20",
31
32     // 学历 (10:研究生 20:本科 30:大专 40:高中 91:初中及以下)
33     "education": "20",
34
35     // 联系信息模块
36     "contact_info": {
37       // 最近5个手机号 (格式:号码(最后使用年月), 反映用户稳定性)
38       "recent_mobiles": [
39         "138****1111(2025-05)",
40         "139****2222(2024-12)",
41         "136****3333(2024-08)",
42         "137****4444(2023-11)",
43         "135****5555(2023-05)"
44       ],
45
```

```

46 // 电子邮箱（用于账单通知）
47 "email": "zhangsan@example.com",
48
49 // 通讯地址（当前有效联系地址）
50 "mailing_address": "北京市海淀区西三环中路XX小区1号楼",
51
52 // 户籍地址（身份证登记地址）
53 "household_address": "河北省石家庄市长安区XX街道"
54 },
55
56 // 配偶信息（仅当marital_status=20时返回有效数据）
57 "spouse_info": {
58     "name": "李四",
59     // 配偶证件类型（同cert_type编码）
60     "cert_type": "1",
61     // 配偶证件号码（脱敏）
62     "cert_number": "11010119900202XXXX",
63     // 配偶工作单位
64     "employer": "北京大学附属第一医院",
65     // 配偶联系电话
66     "contact": "139****8888"
67 }
68 },
69
70 // ===== 防欺诈提示 ===== //
71 "fraud_warnings": {
72     // 是否存在防欺诈标记（true时需人工复核）
73     "has_fraud_alert": true,
74
75     // 防欺诈提示生效日期（YYYY-MM-DD）
76     "alert_start_date": "2025-01-15",
77
78     // 防欺诈提示截止日期（YYYY-MM-DD）
79     "alert_end_date": "2025-12-31",
80
81     // 是否存在异议标注（true表示用户对报告数据有争议）
82     "has_dispute": false
83 },
84
85 // ===== 职业历史 ===== //
86 "employment_history": [
87     {
88         // 工作单位全称
89         "employer": "腾讯科技有限公司",
90
91         // 进入本单位年份（YYYY格式）
92         "start_year": "2023",
93

```

```

94 // 单位性质 (11:机关单位 21:私营企业 31:外资企业 41:事业单位)
95 "company_type": "21",
96
97 // 单位联系电话
98 "company_phone": "010-88889999",
99
100 // 职务名称
101 "position": "高级工程师",
102
103 // 所属行业 (自由文本)
104 "industry": "IT"
105 },
106 {
107     "employer": "阿里巴巴集团",
108     "start_year": "2020",
109     "company_type": "21",
110     "company_phone": "0571-66667777",
111     "position": "技术经理",
112     "industry": "互联网"
113 }
114 ],
115
116 // ===== 居住历史 ===== //
117 "residence_history": [
118     {
119         // 居住地址全称
120         "address": "杭州市西湖区文一西路XX公寓",
121
122         // 住宅固定电话
123         "residence_phone": "0571-55556666",
124
125         // 地址更新时间 (YYYY-MM格式)
126         "update_date": "2023-04"
127     },
128     {
129         "address": "上海市浦东新区张江高科技园区",
130         "residence_phone": "021-33334444",
131         "update_date": "2020-11"
132     }
133 ],
134
135 // ===== 信用概要 ===== //
136 "credit_summary": {
137     // 特殊记录标记 (影响风控决策)
138     "special_records": {
139         // 是否出现过呆账 (true直接拒贷)
140         "has_bad_debt": false,
141

```

```

142 // 是否出现过资产处置
143 "has_asset_disposal": false,
144
145 // 是否出现过保证人代偿（反映还款能力问题）
146 "has_guarantor_repayment": true
147 },
148
149 // 贷款逾期统计（过去5年内）
150 "loan_overdue": {
151     // 发生过逾期的账户数量
152     "count": 2,
153
154     // 所有账户累计逾期月份总数
155     "months": 5,
156
157     // 单月最高逾期金额（单位：元）
158     "max_amount": 8500,
159
160     // 最长连续逾期月数
161     "max_duration": 2
162 },
163
164 // 贷记卡逾期统计
165 "credit_card_overdue": {
166     "count": 1,
167     "months": 3,
168     "max_amount": 12000,
169     "max_duration": 1
170 },
171
172 // 异常状态标记
173 "abnormal_flags": {
174     // 信用卡是否存在呆账/止付状态
175     "has_credit_card_abnormal": false,
176
177     // 贷款是否出现异常五级分类（如次级、可疑、损失）
178     "has_abnormal_loan_class": true
179 },
180 },
181
182 // ===== 授信额度 ===== //
183 "credit_limits": {
184     // 各类型账户最大授信额度
185     "max_limits": {
186         // 非循环贷最高授信（如房贷、车贷）
187         "non_revolving": 500000,
188
189         // 信用卡最高授信额度

```

```

190     "credit_card": 100000,
191
192     // 循环贷最高授信（如信用贷）
193     "revolving": 300000,
194
195     // 循环额度下分账户最高授信
196     "revolving_sub": 200000,
197
198     // 所有账户中的最大授信值
199     "overall_max": 500000
200 },
201
202 // 各类型账户最小授信额度
203 "min_limits": {
204     "credit_card": 20000,
205     "revolving": 50000,
206     "non_revolving": 100000,
207     "revolving_sub": 50000,
208
209     // 消费金融类机构最低放款额度
210     "consumer_finance": 30000
211 },
212
213 // 次高授信额度（用于分析额度分布）
214 "second_max_limits": {
215     "credit_card": 80000,
216     "revolving": 250000,
217     "non_revolving": 300000,
218     "revolving_sub": 150000
219 },
220
221 // 时间窗口统计指标（反映额度变化趋势）
222 "time_window_stats": {
223     // 近3个月指标
224     "last_3m": {
225         "credit_card": {
226             "min": 20000, // 期间最低额度
227             "max": 100000, // 期间最高额度
228             "avg": 60000 // 期间平均额度
229         },
230         "revolving": { "min": 50000, "max": 300000, "avg": 175000 },
231         "consumer_finance": { "min": 30000, "max": 50000, "avg": 40000 }
232     },
233
234     // 近6个月指标（计算规则同近3月）
235     "last_6m": {
236         "credit_card": { "min": 20000, "max": 100000, "avg": 65000 },
237         "revolving": { "min": 50000, "max": 300000, "avg": 180000 }

```

```

238         },
239     },
240     // 近12个月指标（计算规则同近3月）
241     "last_12m": {
242         "credit_card": { "min": 20000, "max": 100000, "avg": 70000 },
243         "revolving": { "min": 50000, "max": 300000, "avg": 190000 }
244     }
245 }
246 }
247 }

```

如果觉得定义Bean太麻烦，可以通过Cursor让他帮你定义，正好也熟悉一下Cursor，AI编辑工具现在也是非常流行。

实现的效果：每次调用这个接口，都能返回不同用户的征信报文，字段名不变，字段值变。

建议生成数值的时候，用抽象工厂来生成，因为这里可以生成字符串、数字、地址、公司名。

要求2

要求建立以下几张表：

1. 任务表，包含任务ID、任务总数、任务状态（初始化、进行中、已完成、失败）、创建时间、更新时间等
2. 任务明细表，包含任务ID、征信报文（取的是第一个要求里面的报文）、任务状态（初始化、进行中、已完成、失败）、创建时间、更新时间
3. 征信报告主表，字段从JSON里面取几个核心的就行，以下几个表都是同样的处理方式。
4. 身份信息表
5. 手机号历史表
6. 职业记录
7. 居住记录
8. 授信额度表

要求：给出详细的DDL定义+具体的ER图，主要看下关联关系对不对。练习一下自己的建模能力。

要求3

服务中接入一个JMQ的依赖，并且实现一个发送消息和接收消息的小Demo就行。

这里有时间的话看下JMQ的事务消息，简单写一写JMQ的事务消息怎么写的。

JMQ看起来是没办法通过Docker部署，找了一圈发现是通过云服务托管的方式提供服务，并且简历中可能得改一下，现在叫JCQ。

不过定价比较便宜，如果觉得麻烦的话可以使用RabbitMQ，Docker里面起一个就行，刚好可以练习一下Docker-Compose。

<https://www.jdcloud.com/cn/products/message-queue>

<https://docs.jdcloud.com/cn/message-queue/price-overview>

要求4

实现以下逻辑如下：

1. 接口1逻辑：创建一个清洗征信报文的任务，写入到任务表，状态为初始化。并且使用随机函数，写入N（N不少于100000）条任务明细，需要调用要求1中的接口获取征信报文，存到任务明细表中，任务状态为初始化。接口返回为任务ID
2. 接口2逻辑：提交任务，传入参数为任务ID，状态从初始化变成进行中。并且遍历任务明细表，将任务明细ID发到JMQ中，这里需要定义Topic。
3. 消费者：监听Topic，消费任务明细ID，查询到对应的报文，通过N-tree算法思路，清洗对应的字段到征信报告主表、身份信息表、手机号历史表、职业记录、居住记录、授信额度表，要求保证事务，并且考虑幂等。
4. 每条明细消费完之后，将明细表中的进行中状态变成已完成，如果失败了变成失败状态。
5. 所有的明细都清洗完成之后，变更任务表中的任务状态，如果明细表中所有的状态都是成功，那么任务表就是成功状态，如果有一个是失败状态，那么主表就是失败状态。

提示一点：判断是否所有的JMQ消息都消费完成，可以使用Redis中的计数器来实现，禁止每次对任务明细表进行Count。

要求5

把你现在的金锐的项目通过Docker的方式部署起来，并且能够通过postman访问即可。

这个比较复杂，估计你会遇到很多问题，有时间可以搞一下，没时间就算了。

提示一下：需要添加Dockerfile文件、建立Docker-compose.yaml文件。

如果Docker搞太麻烦，可以直接部署到你的云服务器里面去，然后开放一个端口，设置IP白名单，支持你http调用即可。

要求6

要求：模拟OOM场景，接入EasyExcel文件导入导出依赖。

写一个接口，将第四步写入到表里面的10万条数据，通过接口的方式导出成Excel，并且将JVM的堆内存调小，检查是否会出现OOM的情况，预期肯定会内存爆掉。

如果出现OOM，通过jstat、jmap导出堆文件查看瓶颈出现在哪里。

要求7

这里没有太多表可以练习的，所以用团购那个项目的SQL来练一下就行。

业务场景：

平台运营后台需要统计某时间段内各团长的业绩排名，显示总销售额、佣金、订单数、商品数，支持按佣金倒序分页。此类报表常用于激励政策、绩效考核、分红结算等业务。

SQL优化大概做了几个之后就差不多有感觉了，知道怎么根据执行计划去优化索引、改写SQL就可以了。

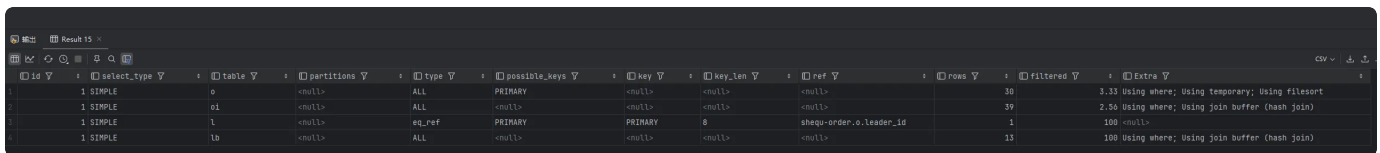
复杂SQL示例：

```

1  SELECT
2      l.id AS leader_id,
3      l.name AS leader_name,
4      COUNT(DISTINCT o.id) AS order_count,
5      SUM(o.total_amount) AS total_sales,
6      SUM(lb.bill_amount) AS total_commission,
7      COUNT(DISTINCT oi.sku_id) AS sku_count
8  FROM `shequ-user`.leader l
9  JOIN `shequ-order`.order_info o
10     ON l.id = o.leader_id
11     AND o.create_time BETWEEN '2024-01-01 00:00:00' AND '2024-06-30 23:59:59'
12     AND o.is_deleted = 0
13  JOIN `shequ-order`.order_item oi
14     ON o.id = oi.order_id
15     AND oi.is_deleted = 0
16  LEFT JOIN `shequ-user`.leader_bill lb
17     ON lb.leader_id = l.id
18     AND lb.bill_type = 'ORDER'
19     AND lb.bill_time BETWEEN '2024-01-01 00:00:00' AND '2024-06-30 23:59:59'
20     AND lb.is_deleted = 0
21  GROUP BY l.id
22  ORDER BY total_commission DESC
23  LIMIT 0, 20;

```

分析执行计划：



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	extra
1	SIMPLE	o	<null>	ALL	PRIMARY	<null>	<null>	<null>	30		3.33 Using where; Using temporary; Using filesort
1	SIMPLE	oi	<null>	ALL	<null>	<null>	<null>	<null>	39		2.50 Using where; Using join buffer (hash join)
1	SIMPLE	l	<null>	eq_ref	PRIMARY	PRIMARY	8	shequ-order.o.leader_id	1	100	<null>
1	SIMPLE	lb	<null>	ALL	<null>	<null>	<null>	<null>	11	100	Using where; Using join buffer (hash join)

用上索引的只有：leader表中的主键ID

要求：尝试分析执行计划，然后通过加减索引，调整SQL，来验证哪种方式能把这个SQL中每个表都能用上索引。

更高要求：看看如果用上索引之后，能不能在索引的基础上继续提高索引的效率，重点看type字段。

多表统计报表型查询，容易出现索引失效、回表、临时表、文件排序等性能瓶颈。

改进方向：设计联合索引、select字段、SQL改写（如分步统计、物化视图）。