

Project intégration de données

Yaroslav Konyshev
Formation ECAM Data Scientist
2018-2019

Fiche de lancement de projet

Projet	
Titre du Projet	Analyse des demandes de crédit pour 2010-2018 années pour Home Credit Bank en Chine
Chef de Projet	Yaroslav Konyshev

Identification	
Type de Projet	Recherche
Programme de rattachement	No
N°référence	1
Projet antérieur	No

Client, maitre d'ouvrage	
Désignation	Home Credit Bank China
Adresse	Futian CBD 518048 Shenzhen, China
Représentant du client	HOME CREDIT A.S. Nové sady 996 / 25 602 00 Brno Czech Republic
Correspondant interne	No

Cadre contractuel			
Mode de consultation	l'analyse initiale		
Origine du financement	Futian CBD 518048 Shenzhen, China		
Type de contrat	Contrat unique		
Montant	4 000 euro	Date de début	26.11.2018
Date de signature	25.11.2018	Date de fin	03.12.2018

Objectif du projet
Concevoir un modèle de données, créer la base de données, importer les données, faire l'analyse initial des données

Planification
Principales phases du projet, tâches et jalons (Work Packages and Milestones)
<p>Phase 1, Creation base de données:</p> <ul style="list-style-type: none"> – Télécharger les données – Importer tous les données dans la base de données – Extraire les données pour le projet (faire transformation) – Supprimer des données supplémentaires – Ajouter des contraintes et des indices <p>Phase 2, Analyse des demandes de crédit:</p> <ul style="list-style-type: none"> – Création de requêtes SQL <p>Phase 3, Présentation des résultats</p>

Organisation	
Service/Département pilote	ECAM Strasbourg
Chef de Service/Département pilote	Yaroslav Konyshev
Co-réalisateurs internes	No
Co-réalisateurs externes	No
Moyens spécifiques liés au projet	No
Comité de pilotage interne	Groupe d'étudiants de formation
Comité de pilotage externe	Home Credit Bank China
Aval du projet	Signature:

Liste des Livrables

- Scripts pour creation base de données
- Structure de la base de données
- Base de données
- Scripts pour l'analyse

Liste des Tâches

Tâches	Charges	Ressources	Dépendances	Coût
Télécharger les données	1 jour	1 développeur	site http://kaggle.com/	500 euro
Importer tous les données dans la base de données	3 jour	1 développeur	RDBMS Environnement, Télécharger les données	1500 euro
Extrait les données pour le projet	1 jour	1 développeur	RDBMS Environnement, Importer tous les données dans la base de données	500 euro
Supprimer des données supplémentaires	0.5 jour	1 développeur	RDBMS Environnement, Extrait les données pour le projet	250 euro
Ajouter des contraintes et des indices	0.5 jour	1 développeur	RDBMS Environnement, Supprimer des données supplémentaires	250 euro
Création de requêtes SQL pour l'analyse	1 jour	1 développeur	RDBMS Environnement, Ajouter des contraintes et des indices	500 euro
Présentation des résultats	1 jour	1 développeur	Création de requêtes SQL pour l'analyse	500 euro
Prix Total				4 000 euro

Tableau de bord

Charge total du projet: **8 j/h**

Durée totale du projet: **8 jours**

Coût de ressources:

1 j/h développement = **250 Euro**

1 jour RDBMS environment = **10 Euro**

Coût total du projet: **2080 Euro**

Bilan de Fin de Projet

Trop de temps a été consacré à la correction. Le projet a été lancé sans une compréhension claire des besoins du client.

Le projet est réalisable en 3 jours aux conditions suivantes:

- forte compréhension des besoins du client
- structure claire du projet avec définition des tables et architecture
- données propres
- utilisation ETL pour le téléchargement de données

Schéma d'Architecture Logique


Dossier avec des fichiers *.csv 

Table name	lines	columns
application_test.csv	48 741	121
application_train.csv	307 511	122
previous_application.csv	1 670 214	37



ETL

- lire le dossier avec les fichiers et insérer le nom des fichiers dans la table temporaire
- lire la première ligne des fichiers, divisée par des virgules et créer des tables avec colonnes du type texte
- lire les données des fichiers csv et remplir les tableaux



DB Prod



Extrait les données pour le projet

- Scripts SQL pour la création de tables finales basées sur les données brutes .
- Transformation de données
- Nettoyage les tables supplémentaires



Transformation finale de la base de données BI

- Convertir les colonnes en un type de données approprié
- Ajouter des index et des contraintes



DB BI



Table name	lines	columns
calendar	4255	6
credit_types	4	2
demande_de_credit	1 670 214	10
achat_types	28	2
client	356 255	4



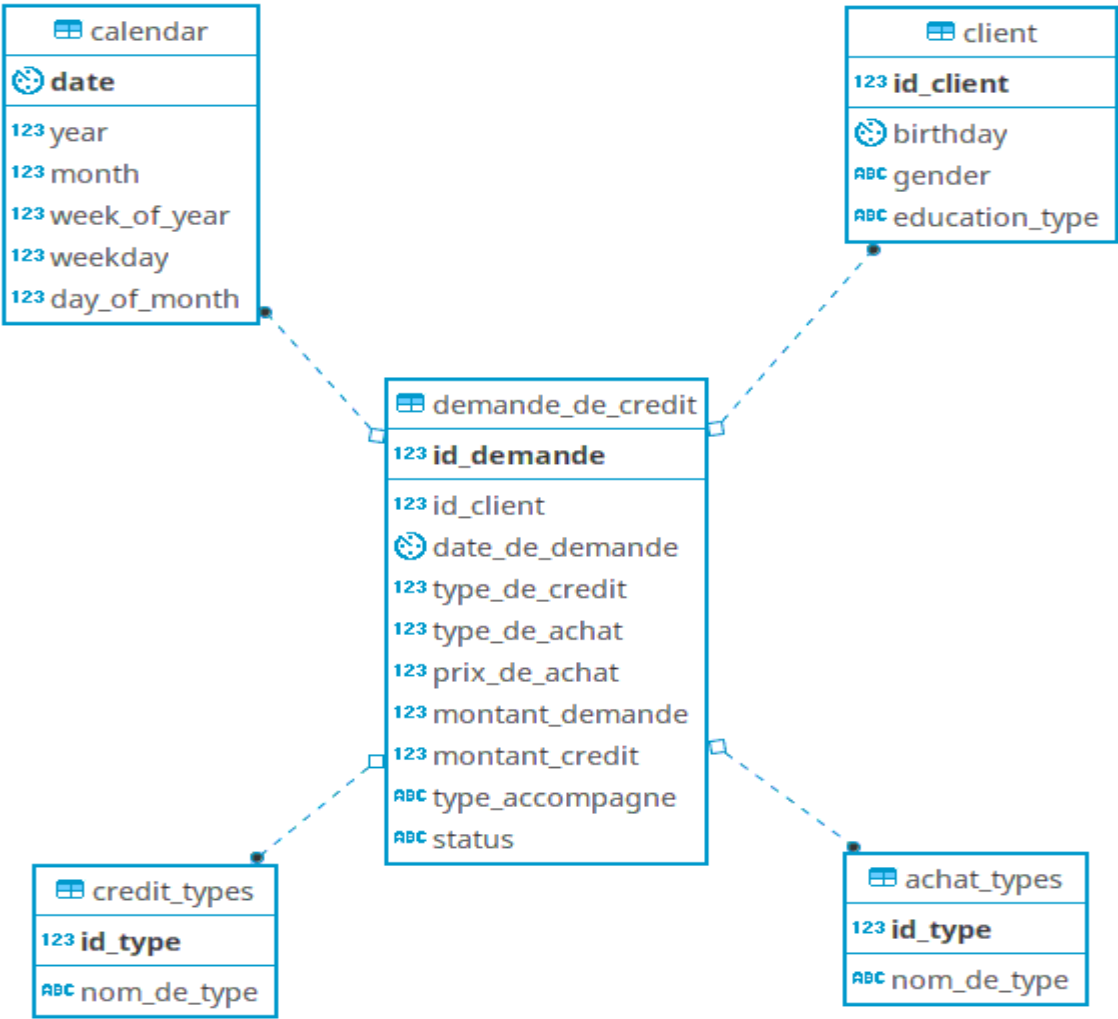
Requêtes SQL



Résultats



MLD



MPD

```
CREATE TABLE calendar (  
    "date" date NOT NULL,  
    "year" int2 NULL,  
    "month" int2 NULL,  
    week_of_year int2 NULL,  
    weekday int2 NULL,  
    day_of_month int2 NULL,  
    CONSTRAINT calendar_pkey null  
)  
  
CREATE TABLE client (  
    id_client int4 NOT NULL,  
    birthday timestamptz NULL,  
    gender text NULL,  
    education_type text NULL,  
    CONSTRAINT client_pkey null  
)  
  
CREATE TABLE credit_types (  
    id_type int2 NOT NULL,  
    nom_de_type text NULL,  
    CONSTRAINT credit_types_pkey null  
)  
  
CREATE TABLE achat_types (  
    id_type int2 NOT NULL,  
    nom_de_type text NULL,  
    CONSTRAINT achat_types_pkey null  
)  
  
CREATE TABLE demande_de_credit (  
    id_demande int4 NOT NULL,  
    id_client int4 NULL,  
    date_de_demande date NULL,  
    type_de_credit int2 NULL,  
    type_de_achat int2 NULL,  
    prix_de_achat float4 NULL,  
    montant_demande float4 NULL,  
    montant_credit float4 NULL,  
    type_accompagne text NULL,  
    status text NULL,  
    CONSTRAINT demande_de_credit_pkey null,  
    CONSTRAINT fk_date_de_demande null,  
    CONSTRAINT fk_id_client null,  
    CONSTRAINT fk_type_de_achat null,  
    CONSTRAINT fk_type_de_credit null  
)
```

Requêtes SQL

1.

```
SELECT
  c.gender ,
  COUNT(d.id_demande) "count" ,
  to_char(PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY prix_de_achat),
    '999 999 999.99') med_of_prix ,
  to_char(MIN(d.montant_demande), '999 999 999.99') "min demande" ,
  to_char(MAX(d.montant_demande), '999 999 999.99') "max demande" ,
  to_char(AVG(d.montant_demande-d.montant_credit),
    '999 999 999.99') "avg (demande - credit)" ,
  to_char(SUM(CASE WHEN d.status = 'Approved' THEN 1 ELSE 0 END)::FLOAT
    / COUNT(d.id_demande)* 100, '99.99 %') "% of approved" ,
  to_char(SUM(CASE WHEN d.status = 'Refused' THEN 1 ELSE 0 END)::FLOAT
    / COUNT(d.id_demande)* 100, '99.99 %') "% of refused"
FROM
  demande_de_credit d
  LEFT JOIN client c ON d.id_client = c.id_client
GROUP BY c.gender;
```

Résultats:

gender	count	med_of_prix	min demande	max demande	avg (demande - credit)	% of approved	% of refused
F	1131886	113 211.00	.00	6 905 160	-21 389.81	62.00 %	17.18 %
M	538273	101 434.50	.00	4 455 000	-19 808.61	62.23 %	17.87 %
XNA	55	180 000.00	.00	1 269 000	-16 301.47	41.82 %	45.45 %

2.

```

WITH age_range AS (
    SELECT a.id_demande,
           CASE
               WHEN a.age_when_demande > 0 AND a.age_when_demande < 25 THEN '0-25'
               WHEN a.age_when_demande >= 25 AND a.age_when_demande < 40 THEN '25-40'
               WHEN a.age_when_demande >= 40 AND a.age_when_demande < 65 THEN '40-65'
               WHEN a.age_when_demande >= 65 THEN '65+'
           END AS range
    FROM (
        SELECT
            d.id_demande,
            EXTRACT(YEAR FROM age(d.date_de_demande,
                                c.birthday))::SMALLINT AS age_when_demande
        FROM
            demande_de_credit d
        LEFT JOIN client c ON d.id_client = c.id_client )a
)
SELECT a.range AS "age when demande range",
       to_char(SUM(CASE WHEN d.type_accompagne = 'Family' THEN 1 ELSE 0 END)::FLOAT
               / COUNT(a.id_demande)* 100,'99.99 %') "% of family",
       to_char(SUM(CASE WHEN d.type_accompagne = 'Group of people' THEN 1 ELSE 0 END)::FLOAT
               / COUNT(a.id_demande)* 100,'99.99 %') "% of group",
       to_char(SUM(CASE WHEN d.type_accompagne = 'Unaccompanied' THEN 1 ELSE 0 END)::FLOAT
               / COUNT(a.id_demande)* 100,'99.99 %') "% of unaccompanied",
       to_char(SUM(CASE WHEN d.type_accompagne = 'Children' THEN 1 ELSE 0 END)::FLOAT
               / COUNT(a.id_demande)* 100,'99.99 %') "%of children",
       to_char(SUM(CASE WHEN d.type_accompagne = 'Spouse, partner' THEN 1 ELSE 0 END)::FLOAT
               / COUNT(a.id_demande)* 100,'99.99 %') "% of spouse"
FROM   age_range a
INNER JOIN demande_de_credit d ON d.id_demande = a.id_demande
GROUP BY a.range
ORDER BY a.range ;

```

Résultats:

age range when demande	% of family	% of group	% of unaccompanied	%of children	% of spouse
0-25	15.14 %	0.33 %	30.44 %	0.24 %	6.30 %
25-40	13.43 %	0.15 %	28.87 %	1.58 %	5.15 %
40-65	12.12 %	0.10 %	31.74 %	2.36 %	2.95 %
65+	6.59 %	0.05 %	27.51 %	0.61 %	0.90 %

3.

```

WITH age_range AS (
    SELECT a.id_demande,
        CASE
            WHEN a.age_when_demande > 0 AND a.age_when_demande < 25 THEN '0-25'
            WHEN a.age_when_demande >= 25 AND a.age_when_demande < 40 THEN '25-40'
            WHEN a.age_when_demande >= 40 AND a.age_when_demande < 65 THEN '40-65'
            WHEN a.age_when_demande >= 65 THEN '65+'
        END AS RANGE
    FROM
        (
            SELECT
                d.id_demande,
                EXTRACT(YEAR FROM age(d.date_de_demande,c.birthday))::SMALLINT AS
age_when_demande
            FROM
                demande_de_credit d
            LEFT JOIN client c ON d.id_client = c.id_client )a
    )
SELECT
    a_t.nom_de_type as "Type de achat",
    SUM(CASE WHEN a_r.range = '0-25' THEN 1 ELSE 0 END) "0-25",
    SUM(CASE WHEN a_r.range = '25-40' THEN 1 ELSE 0 END) "25-40",
    SUM(CASE WHEN a_r.range = '40-65' THEN 1 ELSE 0 END) "40-65",
    SUM(CASE WHEN a_r.range = '65+' THEN 1 ELSE 0 END) "65+",
    COUNT(d.id_demande) "Total"
FROM
    age_range a_r
    INNER JOIN demande_de_credit d ON d.id_demande = a_r.id_demande
    INNER JOIN achat_types a_t ON a_t.id_type = d.type_de_achat
GROUP BY a_t.nom_de_type
ORDER BY "Total" DESC;

```

Résultats:

Type de achat	0-25	25-40	40-65	65+	Total
XNA	31014	333139	570422	16234	950809
Mobile	33757	110730	79746	475	224708
Consumer Electronics	9102	47547	64007	920	121576
Computers	15091	50036	40179	463	105769
Audio/Video	11128	44771	43017	525	99441
Furniture	3083	20340	29718	515	53656
Photo / Cinema Equipment	3933	12898	8146	44	25021
Construction Materials	847	8179	15611	358	24995
Clothing and Accessories	1702	9739	11872	241	23554
Auto Accessories	994	3859	2499	29	7381
Jewelry	718	2771	2733	68	6290
Homewares	211	1354	3333	125	5023
Medical Supplies	38	819	2889	97	3843
Vehicles	310	1519	1521	20	3370
Sport and Leisure	296	1426	1239	20	2981
Gardening	95	968	1578	27	2668
Other	256	1063	1235	0	2554
Office Appliances	358	1142	821	12	2333
Tourism	136	856	662	5	1659
Medicine	44	448	1027	31	1550

Direct Sales	3	65	353	25	446
Fitness	18	115	76	0	209
Additional Service	5	42	80	1	128
Education	30	45	32	0	107
Weapon	13	37	27	0	77
Insurance	14	26	24	0	64
House Construction	0	1	0	0	1
Animals	0	1	0	0	1

4.

SELECT

c."year",
COUNT(d.id_demande)

FROM calendar c

LEFT JOIN demande_de_credit d **ON** c."date" = d.date_de_demande

GROUP BY c."year"

ORDER BY c."year";

Résultats:

year	count
2008	0
2009	0
2010	73 110
2011	99 028
2012	81 955
2013	98 374
2014	126 900
2015	203 421
2016	357 152
2017	585 369
2018	44 905
2019	0
2020	0

5.

```
SELECT
    c.weekday,
    COUNT(d.id_demande)
FROM calendar c
    LEFT JOIN demande_de_credit d ON c."date" = d.date_de_demande
GROUP BY c.weekday
ORDER BY c.weekday;
```

Résultats:

weekday	count
1	232911
2	238120
3	260319
4	242139
5	231880
6	232088
7	232757