Stage ingénieur 3ème année – Promotion 2022-2027		
Sybella SA	**Création d'une plateforme de backtesting**	
Maximilien ROUSSEAU		
Maitre de stage : Vilayphone Nguyen	**Tuteur pédagogique :** NDAO Amath	
Dates du stage : 23 juin – 22 août		

Remerciements

Je tiens à exprimer mes sincères remerciements à la direction de Sybella SA pour m'avoir offert l'opportunité d'effectuer mon stage de troisième année d'ingénieur au sein de leur structure. Mes remerciements vont tout particulièrement à mon maître de stage pour son encadrement, son support et ses conseils précieux durant toute la durée de mon stage.

1. Introduction

Mon stage s'est déroulé du 23 juin au 22 août 2025, soit une durée de huit semaines, au sein de la société Sybella SA.

J'ai choisi ce stage car il représentait pour moi une opportunité d'approfondir mes compétences informatiques – en particulier dans le développement Python et la manipulation de données – tout en m'ouvrant à un domaine que je ne connaissais pas encore : le monde de la finance.

Le sujet initial confié portait sur la création d'une plateforme de backtesting avec une composante d'intelligence artificielle. Après avoir passé un peu moins de la moitié des deux mois à défricher le sujet puis à jalonner le projet, par manque de temps et en raison des difficultés techniques rencontrées, nous avons consacré l'essentiel du dernier mois à la création d'un data lake : sans socle de données fiable, un backtest n'est ni auditable ni crédible ni reproductible. La suite du rapport décrit ces deux phases, la méthode mise en place et ce que j'en ai tiré en tant qu'étudiant-ingénieur.

Ce rapport présente d'abord le contexte et la problématique à l'origine du projet, puis les différentes phases de montée en compétences qui m'ont permis de m'approprier les concepts financiers et les outils logiciels. Il détaille ensuite la réflexion menée autour de la question de la décorrélation des stratégies, la mission technique réalisée, ainsi que les résultats obtenus et les perspectives pour l'entreprise et pour ma formation d'ingénieur.

2. Contexte et problématique

La société Sybella SA est une jeune entreprise dont l'activité couvre principalement le conseil en économie, en organisation d'entreprise et en innovation technologique. Dans ce cadre, elle s'intéresse particulièrement au domaine de la finance quantitative, et plus spécifiquement au développement d'outils d'aide à la décision appliqués à la gestion de portefeuilles d'investissement et au trading systématique.

A noter que tout au long du stage, j'ai reçu des instructions pour utiliser au maximum les outils IA (principalement ceux d'OpenAI mais aussi Claude et Gemini) et que j'ai pu comprendre comment faire de l'IA un outil de productivité en restant maitre de mon projet. J'ai bien compris qu'on a les réponses que la qualité de nos questions mérite.

L'entreprise a exprimé le besoin de disposer d'un outil interne de backtesting, c'est-à-dire d'une plateforme permettant de tester et d'évaluer des stratégies d'investissement et de trading sur des données historiques. L'objectif était de s'affranchir des solutions propriétaires de type « boîtes noires », souvent opaques et coûteuses, afin de garantir une transparence totale dans la construction des signaux, l'évaluation des performances et la gestion des risques.

Les principaux enjeux identifiés étaient les suivants :

- Transparence et auditabilité : chaque résultat doit pouvoir être justifié, tracé et reproduit.
- Reproductibilité et traçabilité : rejouer une expérience, comparer des variantes de signaux et documenter les choix doit être possible à tout moment.
- Maîtrise des coûts : prise en compte des frais de transaction, du slippage (c'est-à-dire l'écart de cours d'exécution d'un ordre souhaité et le réalisé).
- Pérennité des données : sécurisation et gouvernance d'un patrimoine de données financières consolidé.
- Gestion des risques : intégration de mécanismes de contrôle tels qu'une volatilité-cible, des plafonds de contribution par stratégie et des règles de coupure automatique.

Le projet devait ainsi reposer sur :

- Un socle de données fiables (par exemple NorgateData, permettant d'obtenir des prix ajustés *Total Return* et l'historique complet des titres),
- Une architecture Python évolutive s'appuyant sur des bibliothèques modernes (VectorBT Pro),
- Et une modélisation réaliste des coûts de transaction afin de rapprocher les backtests de conditions d'exécution réelles.

Problématique :
Comment concevoir une plateforme de backtesting transparente, reproductible et économiquement viable pour Sybella SA, en alternative aux solutions propriétaires « boîtes noires » ?

3. Montée en compétences

3.1 Lectures de référence

Avant d'aborder la phase technique du projet, une première étape a consisté à acquérir une compréhension plus approfondie du fonctionnement des stratégies quantitatives en finance. Deux ouvrages ont servi de support principal à cette montée en compétences : *Quantitative Momentum* de Wesley R. Gray et *The Acquirer's Multiple* de Tobias Carlisle.

Quantitative Momentum – Wesley R. Gray

Cet ouvrage présente la stratégie du momentum, qui consiste à parier sur la poursuite des tendances passées des prix. L'auteur montre qu'en observant l'évolution des actions américaines sur les 12 derniers mois (hors dernier mois), il est possible de sélectionner les titres les plus performants et de les regrouper dans un portefeuille. Investir chaque mois dans le décile supérieur des actions selon ce critère a permis historiquement (1927–2014) de battre l'indice S&P 500.

Gray insiste sur plusieurs points essentiels :

- Le momentum n'est pas équivalent au style *growth* : il ne se base pas sur les comptes ou les perspectives de croissance, mais uniquement sur l'évolution des prix.
- Combiné à la value (acheter les titres décotés), le momentum présente une corrélation négative (≈ –0,3), ce qui permet d'améliorer la diversification.
- Des améliorations simples existent, comme le filtre *Frog-in-the-Pan* (privilégier les hausses régulières plutôt que les bonds ponctuels) ou l'ajustement saisonnier (réduire l'exposition en janvier, mois historiquement défavorable au momentum).
- Le risque majeur reste le crash momentum (exemple : mars 2009), contre lequel l'auteur propose deux "airbags" : un coupe-circuit basé sur la moyenne mobile à 200 jours et une diversification accrue (50 à 100 lignes à poids égal).
- Enfin, l'importance de maîtriser les coûts de transaction et la fiscalité est soulignée : la rentabilité du momentum n'existe que si l'on minimise les frictions.

En résumé, *Quantitative Momentum* démontre que la tendance des prix peut constituer une source de performance robuste, mais seulement si elle est accompagnée de discipline, de diversification et d'une gestion rigoureuse des coûts.

The Acquirer's Multiple – Tobias Carlisle

L'approche proposée par Carlisle est à l'opposé de celle de Gray : il s'agit d'une stratégie value contrarienne. L'investisseur cherche à profiter du retour à la moyenne en achetant les actions que le marché délaisse fortement.

L'indicateur central est l'Acquirer's Multiple, défini comme le rapport entre la valeur d'entreprise (Enterprise Value) et le résultat d'exploitation (EBIT). Plus ce ratio est faible, plus l'action est considérée comme sous-évaluée. Chaque année, on sélectionne un portefeuille concentré (20 à 30 titres) parmi les actions américaines présentant les multiples les plus bas.

Les points clés de cette stratégie sont :

- Elle a historiquement surperformé le marché sur longue période (tests 1973–2017), mais elle implique d'accepter des phases de pertes importantes (–50 % ou plus).
- Le principal risque est celui des value traps (entreprises bon marché en apparence mais structurellement fragiles). Carlisle recommande donc d'exiger un bilan solide et de diversifier sur une vingtaine de lignes au minimum.
- Cette approche est peu coûteuse : rééquilibrée une fois par an, elle est adaptée aux investisseurs long terme.
- Elle est faiblement corrélée au momentum (≈ –0,3), ce qui rend les deux stratégies complémentaires.

Ainsi, là où Gray mise sur la poursuite des tendances positives, Carlisle cherche à exploiter l'exagération des mouvements négatifs. Leur combinaison dans un portefeuille équilibré permet de réduire la volatilité et les pertes maximales, tout en maintenant un rendement moyen attractif.

Enseignements pour le stage

La lecture croisée de ces deux ouvrages m'a permis de mieux comprendre la logique de construction des facteurs quantitatifs et l'intérêt de les combiner. Momentum et Value, bien qu'opposés dans leur philosophie, sont complémentaires dans un portefeuille multi-stratégies.

Cette réflexion a constitué une base utile pour la suite du stage, notamment pour l'utilisation de bibliothèques de backtesting comme VectorBT Pro, qui permettent de modéliser et tester ce type de stratégies à grande échelle.

Même si ces stratégies peuvent sembler éloignées de la pratique de mon stage, leur étude m'a permis de comprendre les logiques fondamentales de la finance quantitative et de mieux appréhender les choix de conception d'un outil de backtesting.

3.2. Tutoriels VectorBT® PRO

Afin de me familiariser avec les outils modernes de backtesting et de simulation en finance quantitative, j'ai travaillé avec **VectorBT Pro**, une bibliothèque Python professionnelle spécialisée dans la modélisation et le test de stratégies d'investissement à grande échelle. Disposer d'une licence professionnelle m'a permis d'accéder à l'ensemble des fonctionnalités avancées et à une série de tutoriels structurés, que j'ai suivis de manière progressive.

Tutoriels réalisés

J'ai complété cinq tutoriels proposés dans la documentation VectorBT Pro :

1. **Basic RSI** : mise en œuvre d'une stratégie simple basée sur l'indicateur de force relative (*Relative Strength Index*), pour comprendre la logique des signaux techniques.
2. **Stop Signals** : intégration de règles de coupe-circuit et de protection, permettant de tester l'impact de différents stops (stop-loss, take-profit, trailing stop).
3. **Pairs Trading** : mise en œuvre d'une stratégie de convergence statistique sur deux actifs corrélés, afin d'explorer la logique d'arbitrage et de *mean reversion*.
4. **Portfolio Optimization** : application de méthodes quantitatives d'optimisation de portefeuilles (allocation de risques, optimisation de Sharpe, etc.).
5. **Signal Development** : génération, test et comparaison de différents signaux d'entrée et de sortie. Ce tutoriel m'a particulièrement marqué, car il illustre la démarche systématique de création d'indicateurs quantitatifs et de backtests reproductibles.

Focus sur le tutoriel *Signal Development*

Le tutoriel *Signal Development* m'a semblé le plus formateur car il illustre la chaîne complète permettant de passer d'une intuition de marché à un signal exploitable par un simulateur.

Dans VectorBT Pro, un signal est représenté par un masque booléen (True/False) plutôt qu'un ordre déjà dimensionné. Cette approche simple permet de comparer plusieurs variantes dans des conditions strictement identiques (mêmes données, mêmes hypothèses de frais et de slippage), ce qui réduit fortement le risque d'erreurs expérimentales.

Plusieurs points méthodologiques se sont révélés déterminants :

- **Robustesse** : éliminer tout risque de *look-ahead* (erreur courante qui consisterait à utiliser une information future), assainir les valeurs manquantes (NaN) et garantir l'alignement temporel.
- **Gestion des croisements** : grâce au concept de *partitions*, on évite les rafales de signaux inutiles lorsqu'un indicateur croise un seuil.
- **Sémantique des signaux** : possibilité de travailler avec des masques simples (entrée/sortie) ou détaillés (entrée/sortie long et short), selon le degré de contrôle souhaité.
- **Hygiène expérimentale** : séparer la génération des signaux (logique théorique) de leur simulation (mise en portefeuille), afin d'éviter toute ambiguïté et de comparer différentes idées dans un cadre identique.

En pratique, ce TP m'a appris à industrialiser la fabrication d'un signal : partir d'une condition humaine lisible, la traduire en un masque vectorisé robuste, la décliner sur une grille de paramètres, puis l'acheminer vers un simulateur de portefeuille sans conflits logiques.

Concrètement, cette démarche outille la suite du stage : on travaille avec des signaux reproductibles et auditables, plutôt qu'avec des backtests "optimistes".

Note

Les notes techniques détaillées sur la génération de signaux avec VectorBT Pro (gestion des partitions, logique d'alignement, exemples de code et de masques booléens) sont présentées en Annexe. Elles reprennent de manière exhaustive le processus suivi et les précautions mises en œuvre pour assurer la validité expérimentale.

Au-delà de l'aspect technique, ce travail de montée en compétences m'a permis de mieux saisir la complexité de la construction de stratégies quantitatives et de comprendre pourquoi la rigueur méthodologique est indispensable. Cette réflexion a préparé le terrain pour la suite du stage, consacrée à l'étude de la robustesse des systèmes de trading et à la recherche de décorrélation.

4. Recherche et réflexion personnelle

Au cours du stage, une question de recherche (sans doute pour me faire mieux comprendre son objet) m'a été posée :
« Peut-on encore gagner avec des systèmes décorrélés en 2025 ? »

Cette interrogation est essentielle dans le monde du trading quantitatif. La diversification repose sur l'idée que combiner plusieurs stratégies faiblement corrélées permet de lisser les performances et de réduire le risque global. Or, avec l'augmentation de la concurrence et la diffusion rapide des idées, de nombreux signaux historiques ont vu leur efficacité diminuer.

4.1. Constats généraux

Il est encore possible de générer de la performance, mais pas en comptant sur une « idée miracle ». Les résultats proviennent désormais de l'accumulation de plusieurs petits avantages statistiques (ou *edges*), chacun modeste mais réel, que l'on combine intelligemment. L'essentiel est de maximiser leur indépendance effective. Deux stratégies construites sur la même logique ou le même horizon ne sont pas vraiment deux paris distincts, mais une simple répétition du même pari.

Un autre constat est que les corrélations augmentent en période de stress : au moment où l'on a le plus besoin de diversification, celle-ci s'érode. La conséquence est qu'il faut tester et calibrer les portefeuilles non seulement avec des corrélations « moyennes », mais aussi avec des corrélations de crise (ou *downside correlations*), observées lors des pires périodes de marché.

Enfin, il est indispensable de tenir compte des coûts d'exécution : spread, impact de marché, frais. Un edge de quelques points de base par trade disparaît immédiatement si la stratégie tourne trop vite ou sur des tailles trop grandes par rapport au volume échangé.

4.2. Tester proprement les stratégies

La discipline méthodologique est centrale. Tester une anomalie ou un signal quantitatif suppose de :

- utiliser des bases de données complètes incluant les radiations (afin d'éviter le biais de survivance),
- respecter un calendrier d'information réaliste (éviter le *look-ahead*),
- appliquer des méthodes statistiques robustes (t-statistics corrigées de l'autocorrélation, seuils de significativité plus stricts pour éviter les faux positifs).

Lorsque l'on applique ces standards, le fameux « zoo » d'anomalies documenté par la recherche académique se vide largement : beaucoup de signaux ne résistent pas à des tests robustes ou disparaissent une fois les coûts intégrés. Mais quelques briques solides subsistent.

4.3. Les briques robustes : trend et carry

Deux styles d'investissement se distinguent par leur persistance dans le temps et leur robustesse :

- Trend following (momentum temporel) : exploiter la tendance passée d'un actif (positif ou négatif) pour anticiper une poursuite.
- Carry : encaisser un rendement prévisible à l'avance (différentiel de taux en devises, rendement courant et pente de courbe en obligations, roll yield en matières premières).

Ces briques, bien que modestes, sont reproductibles et codables proprement. Elles sont complémentaires : la tendance réagit aux grands mouvements directionnels, tandis que le carry fonctionne mieux dans des régimes stables.

4.4. Cyclicité et gouvernance du risque

Ces stratégies connaissent des phases fastes et des périodes difficiles (*cyclicité*). Par exemple, le trend following performe bien lors de grandes tendances directionnelles (hausse du dollar, pétrole durablement élevé), mais souffre dans des marchés sans direction claire.

Pour gérer cette cyclicité, plusieurs règles de gouvernance sont nécessaires :

- désynchroniser les signaux (ne pas tout exécuter le même jour/horaire),
- mélanger plusieurs briques (trend, carry, autres signaux lents),
- dimensionner le risque avec une volatilité-cible (par ex. 10–12 % annuels),
- appliquer des mécanismes de coupure automatique (*kill-switch*) si une brique dépasse un drawdown ou une perte attendue.

4.5. Intégration des corrélations de stress

Un point central est d'intégrer les corrélations de stress dans le dimensionnement des portefeuilles.
Concrètement, on construit deux matrices de corrélations :

- une matrice « normale », estimée sur les données récentes,
- une matrice « de stress », où les corrélations sont plus élevées (par ex. mesurées sur les pires 20 % de mois).

On dimensionne ensuite le portefeuille pour que même dans ce scénario défavorable, les pertes restent compatibles avec les limites fixées (par exemple un *Expected Shortfall* maximum).

4.6. Conclusion de la réflexion

En résumé, il est encore possible en 2025 de gagner avec des systèmes décorrélés, mais cela ne repose plus sur une anomalie unique ou une stratégie miracle. La performance durable vient d'une ingénierie rigoureuse :

- accumuler plusieurs petites briques robustes,
- maximiser leur indépendance effective,
- modéliser les coûts ex-ante et mesurer le *shortfall* ex-post,
- intégrer les corrélations de stress dans la gestion du risque,
- appliquer des garde-fous opérationnels (volatilité-cible, plafonds de risque, kill-switch).

Cette réflexion, bien qu'indépendante de la partie codage, a directement nourri le projet de stage : elle montre pourquoi il est essentiel de bâtir une plateforme de backtesting qui soit transparente, reproductible et économiquement viable, et qui permette de tester la robustesse réelle des stratégies plutôt que de se fier à des backtests trop optimistes.

5. Mission technique : de l'objectif initial à la réalité

5.1. Mission initiale

La mission qui m'avait été confiée au départ consistait à créer une plateforme de backtesting intégrant une composante d'intelligence artificielle. L'outil devait permettre de tester des stratégies d'investissement sur données historiques, en intégrant les coûts de transaction et des mécanismes réalistes de gestion du risque.

Cependant, dès les premières phases de travail, il est apparu que la construction d'une telle plateforme nécessitait en premier lieu un accès efficace à un socle de données solide. La qualité, la cohérence et la traçabilité des données sont en effet des conditions indispensables pour obtenir des backtests transparents, répétables et auditables.

C'est pourquoi la première étape fondamentale du projet a été la conception d'un data lake de données financières robuste, capable de servir de base à tout développement futur de backtesting et d'IA.

5.2. La construction d'un datalake

L'enjeu principal était de créer un système capable de :

- ingérer quotidiennement les données de marché (NorgateData) en moins d'une heure,
- gérer les corrections historiques (splits, reclassifications, révisions),
- permettre le time-travel grâce à des snapshots versionnés,
- garantir la reproductibilité et la sécurité du patrimoine de données.

Après plusieurs expérimentations (Iceberg, Spark), la solution retenue a été basée sur PostgreSQL, en utilisant un modèle de données SCD2 (Slowly Changing Dimension Type 2) pour conserver toutes les versions des séries temporelles et assurer une traçabilité complète.

5.3. Étapes du projet

5.3.1. Première itération : Iceberg local (PyIceberg 0.9.1)

La première piste explorée fut l'utilisation d'Apache Iceberg via la bibliothèque PyIceberg.

- Problèmes rencontrés :
    - Incompatibilité des timestamps en nanosecondes (ns) de Pandas avec le support limité d'Iceberg (us).
    - Difficultés avec l'intégration MinIO (S3 local) sous Windows : gestion des policies, accès HeadObject, configuration du *path-style access*.
    - Performances insuffisantes : ingestion initiale sur gros volumes bien supérieure à une heure.

Conclusion : la solution fonctionnelle mais trop lente → pivot nécessaire.

5.3.2. Deuxième itération : Spark et Hive

Une deuxième tentative a été réalisée avec Spark/Hive en local pour accélérer l'ingestion : stack lourde à opérer sous Windows (dépendances, metastore) et instabilité lors des audits (erreurs mémoire). Malgré de nombreuses tentatives de paramétrage (partitions, compression), l'ensemble restait coûteux au quotidien. On a donc abandonné cette piste pour privilégier une solution plus simple et prévisible en exploitation.

5.3.3. Troisième itération : PostgreSQL avec SCD2 (stabilisée)

La solution finalement retenue fut la plus robuste : un datalake basé sur PostgreSQL.

- Modèle SCD2 : chaque enregistrement est versionné avec des colonnes valid_from, valid_thru et is_active. Les données historiques sont conservées et les snapshots permettent un véritable *time-travel*.
- Ingestion initiale (full-load) :
    - Multiprocessing par année (partitions annuelles), multithreading par symboles.
    - Utilisation massive de la commande COPY ... FROM STDIN (CSV streamé) pour accélérer l'insertion.
    - Suppression temporaire des index avant ingestion, puis recréation a posteriori.

- Mises à jour incrémentales :
    - *lookback paramétrable* (p. ex. 7–14 jours) pour capturer les corrections qui ont nécessairement un impact sur la première ligne de l'historique.
    - Comparaison par source_hash des colonnes métier pour détecter les changements.

- Volume de données : environ 80 millions de lignes ingérées, avec une latence inférieure à une heure pour une mise à jour complète, 1 million de lignes nouvelles quotidiennement compte tenu du versionning d'enregistrement.

Concrètement, PostgreSQL SCD2 s'est révélé plus simple à maintenir, plus lisible à auditer et suffisant en performances pour notre usage (ingestion initiale et incrémentale).
Côté Python les traitements sont massivement multi process (utilisation de tous les cores pour le traitement intensif) et multi thread (pour optimiser les accès IO)

5.4. Résultats obtenus

Les livrables principaux sont :

- Un datalake PostgreSQL structuré, basé sur SCD2, couvrant les séries temporelles de marché.
- Un système de snapshots permettant d'accéder à l'état exact des données à une date donnée (*time-travel*).
- Un ensemble de scripts d'administration :
    - initialisation, reset, migrations versionnées, backup/restore robuste.
    - ingestion initiale (03_run_full.py), mise à jour incrémentale (03_run_incr.py), synchronisations référentiels (04_sync_metadata.py, 05_sync_fundamentals.py, 06_sync_index_memberships.py)..
- Des performances stabilisées : ingestion < 1h, mises à jour incrémentales fiables, cohérence temporelle garantie par un timestamp serveur unique.

5.5. Difficultés rencontrées

Plusieurs difficultés techniques ont marqué ce projet :

- Compatibilités logicielles : PyIceberg 0.9.1 avec Pandas (ns vs us), Java 11 vs 17 pour Spark, binaires PostgreSQL version 17 avec serveur 16 (erreurs GUC transaction_timeout).
- Performances : lenteurs des opérations upsert dans Iceberg, OOM avec Spark, nécessité de drop/recreate d'index pour accélérer PostgreSQL.
- Cohérence temporelle : assurer qu'un seul run_ts serveur soit appliqué partout (symbols, candles, snapshots).
- Sauvegardes : fiabilisation du backup/restore avec pg_dump en s'assurant que la version des binaires corresponde à celle du serveur.

5.6. Bonnes pratiques et enseignements

Au fil du projet, plusieurs leçons opérationnelles ont été dégagées :

- Ne jamais mélanger des binaires PostgreSQL d'une version différente de celle du serveur.
- Préférer des solutions simples et robustes (PostgreSQL + SCD2) à des architectures complexes mais instables (Iceberg + Hive sur Windows).
- Exploiter le partitionnement par année pour paralléliser l'ingestion et réduire la contention.
- Gérer les index avec discernement (drop/recreate au bon moment) pour accélérer l'ingestion massive.

Enfin, ce travail m'a montré l'importance d'un datalake fiable et traçable comme prérequis à tout développement de plateforme de backtesting. L'IA ou les stratégies complexes n'ont de sens que si elles reposent sur des données propres, versionnées et reproductibles.

6. Résultats et apports

6.1. Apports pour l'entreprise

Le projet a permis à **Sybella SA** de disposer d'un socle technique robuste pour ses futurs travaux de recherche et de développement en finance quantitative :

- **Mise en place d'un datalake structuré** : toutes les données de marché issues de NorgateData sont désormais stockées dans une base PostgreSQL avec versioning SCD2, permettant le *time-travel* et la traçabilité complète.
- **Fiabilité et reproductibilité** : chaque mise à jour est horodatée, contrôlée et auditable, ce qui garantit la cohérence et la qualité des données.
- **Performance** : les scripts d'ingestion et de mise à jour atteignent les objectifs de vitesse (full-load < 1 heure, mises à jour incrémentales rapides).
- **Outils administratifs** : l'entreprise dispose désormais de procédures robustes pour initialiser la base, gérer les migrations, effectuer des sauvegardes/restaurations et valider les données avant écriture.
- **Base pour le backtesting futur** : ce datalake constitue un prérequis indispensable pour développer ensuite la plateforme de backtesting transparente et auditable, en alternative aux solutions propriétaires « boîtes noires ».

En résumé, même si la mission initiale (plateforme complète de backtesting avec IA) n'a pas été atteinte, l'entreprise bénéficie aujourd'hui d'un **patrimoine de données structuré et durable**, qui servira de fondation à la suite du projet.

6.2. Apports pour l'étudiant

Ce stage m'a fait progresser sur trois plans. Sur le plan technique, j'ai renforcé le développement en Python (scripts multiprocessus et multithread pour l'ingestion et la gestion de bases), la manipulation avancée de données financières (intégration NorgateData et contrôles de qualité OHLCV), la conception d'un modèle de données versionné (SCD2) et

l'exploitation de PostgreSQL à grande échelle, tout en mobilisant des frameworks spécialisés comme VectorBT Pro, PyIceberg et Spark. Sur le plan méthodologique, j'ai conduit le projet de manière incrémentale en assumant des pivots (Iceberg → Spark → PostgreSQL) et en appliquant une rigueur scientifique — vérification systématique, reproductibilité, absence de look-ahead — tout en testant, documentant et installant des garde-fous. Enfin, sur le plan humain, j'ai découvert la finance quantitative, me suis adapté à un environnement de start-up avec un rôle central et une forte autonomie, dans un dialogue direct avec mon maître de stage.
J'ai également renforcé ma capacité à travailler en équipe et pu apprécier tout l'intérêt du pair programming.

7. Analyse critique et perspectives

7.1. Écart entre mission initiale et mission réalisée

La mission initiale portait sur la création d'une plateforme de backtesting intégrant une composante d'intelligence artificielle. Dans les faits, cet objectif n'a pas pu être atteint dans le temps imparti du fait des difficultés rencontrées dans la construction d'un datalake robuste pour consolider et versionner les données financières.

Cet écart illustre bien la réalité des projets techniques : il est souvent nécessaire de résoudre en priorité les problèmes de fondations avant de pouvoir attaquer la partie visible et applicative. Dans ce cas, l'absence d'un socle de données fiable aurait rendu tout backtesting bancal et peu crédible.

Ainsi, même si le livrable final diffère de la mission initiale, il constitue un résultat essentiel et directement réutilisable.

7.2. Enseignements tirés

Plusieurs enseignements ont marqué cette expérience :

- La donnée est la pierre angulaire : un projet quantitatif ne peut réussir que si les données sont propres, traçables et versionnées.
- La simplicité robuste prime sur la complexité fragile : après des tentatives avec Iceberg et Spark, la solution PostgreSQL + SCD2 s'est révélée plus simple à mettre en place, mais surtout plus fiable et performante.
- L'importance de la reproductibilité : les mécanismes de snapshots, de vérification et de cohérence temporelle ont garanti une traçabilité indispensable.
- La nécessité d'adapter le projet : plutôt que de forcer la mission initiale, j'ai appris à pivoter vers une solution pragmatique et réaliste, en accord avec les besoins réels de l'entreprise.

7.3. Limites du projet

Malgré les avancées, certaines limites subsistent :

- Le **logiciel de backtesting complet** n'a pas encore été réalisé ; il reste à bâtir l'interface utilisateur, les modules de simulation et d'analyse, ainsi que la couche IA.
- L'infrastructure est encore **locale** (Windows + PostgreSQL) : une industrialisation future pourrait nécessiter un déploiement cloud et des mécanismes de haute disponibilité.
- Le **scope de données** reste centré sur NorgateData ; l'intégration d'autres sources (macro, fondamentales enrichies, flux temps réel) pourrait renforcer la pertinence du datalake.

7.4. Perspectives d'évolution

Plusieurs pistes d'amélioration sont envisageables pour prolonger ce travail :

- Finaliser la plateforme de backtesting en s'appuyant sur le datalake : intégration de VectorBT Pro ou d'autres frameworks, mise en place de stratégies multi-facteurs, backtests reproductibles avec reporting standardisé.
- Ajouter une couche d'IA : par exemple, utiliser des modèles de machine learning pour la sélection de signaux ou la détection de régimes de marché.
- Renforcer l'automatisation : scheduler des mises à jour quotidiennes, mettre en place des tests unitaires de qualité de données et des alertes en cas d'anomalie.
- Étendre l'architecture : explorer des solutions de type DuckDB, Polars ou bases distribuées (Snowflake, BigQuery) si le volume de données venait à croître.
- Améliorer la gouvernance : ajout de vues matérialisées pour des agrégats (jour/semaine), index spécialisés pour accélérer les requêtes analytiques, gestion fine des droits d'accès.

8. Conclusion générale

Ce stage de huit semaines au sein de Sybella SA a représenté une expérience riche et complète. D'un point de vue technique, il m'a permis de découvrir la finance quantitative et d'approfondir mes compétences en Python, data engineering et gestion de bases de données. L'évolution du projet, de la conception d'un outil de backtesting vers la mise en place d'un datalake versionné sous PostgreSQL, m'a montré l'importance des fondations de données dans tout projet quantitatif. Sur le plan organisationnel, il m'a permis de renforcer mes aptitudes de travail au sein d'une équipe, ce qui a été essentiel pour pouvoir valoriser mon apport, car si j'avais voulu travailler seul, j'aurai vite été arrêté par mon manque d'expérience. En cherchant à m'intégrer le plus possible mes contributions si modestes soient elles ont été utiles.

Sur le plan méthodologique, j'ai appris à expérimenter différentes approches, à analyser leurs limites et à pivoter vers une solution pragmatique. Cette démarche m'a sensibilisé à la

valeur de la reproductibilité et de la traçabilité, mais aussi à l'importance d'une approche incrémentale et documentée.

Enfin, sur le plan humain, travailler dans une structure jeune m'a permis de gagner en autonomie, organisation et communication technique. Les échanges quotidiens avec mon maître de stage m'ont aidé à progresser et à développer un esprit critique face aux choix techniques et méthodologiques.

En conclusion, ce stage a pleinement répondu à mes attentes : il m'a apporté des compétences techniques solides, une méthodologie d'ingénieur, et une meilleure compréhension du rôle que je souhaite jouer à l'interface entre data engineering et finance quantitative, même si je regrette de ne pas avoir pu abordé les sujets IA.

Résumé / Abstract

Ce stage de 8 semaines au sein de Sybella SA avait pour objectif la création d'un logiciel de backtesting intégrant de l'intelligence artificielle. La mission a finalement évolué vers la mise en place d'un datalake financier robuste, basé sur PostgreSQL et un modèle SCD2, afin de garantir la traçabilité, la reproductibilité et la performance des données de marché. Ce travail constitue un socle technique essentiel pour les futurs développements de la société en finance quantitative.

This 8-week internship at Sybella SA initially aimed at developing a backtesting software with artificial intelligence. The mission eventually evolved towards building a robust financial data lake, based on PostgreSQL and an SCD2 model, to ensure traceability, reproducibility, and performance of market data. This work now provides the technical foundation for the company's future developments in quantitative finance.

- *Français : Backtesting, Finance quantitative, Data engineering, PostgreSQL, NorgateData*
- *Anglais : Backtesting, Quantitative finance, Data engineering, PostgreSQL, NorgateData*

Annexes

Annexe A — Génération de signaux : méthode agnostique & pont VectorBT Pro

Objectif — Décrire une méthode reproductible pour générer des signaux (entrées/sorties) sous forme de masques booléens, garantir l'hygiène expérimentale (sans regard vers le futur), et montrer l'équivalent sous VectorBT Pro (v2025.7.27) pour bénéficier du broadcasting, des utilitaires de partitions/croisements et des générateurs d'exits.

A.0 — Invariants & conventions

- Un signal = un masque booléen aligné sur l'index temporel (True = déclencher, False = ne rien faire).
- Aucune fuite d'information : toute décision à t utilise exclusivement des données disponibles à t.
- Index trié, sans doublon ; NaN de tête maîtrisés (buffer initial) ; colonnes cohérentes entre séries.
- Comparaisons équitables : mêmes hypothèses d'exécution (frais, slippage, timing) pour toutes les variantes.
- Journalisation : paramètres, signature de données (source, as-of), identifiant de run.

Partie I — Méthode agnostique (pandas-first)

I.1 — Hygiène des données (index, NaN, duplications)

Listing A.1 — Hygiène minimale (index, NaN, buffer)

```
import pandas as pd
import numpy as np

# df_ohlc : DataFrame avec colonnes ['Open','High','Low','Close']
df = df_ohlc.copy()
df = df[~df.index.duplicated(keep='last')].sort_index()

# Anti-artefacts après les premières valeurs non-NaN
if df.isna().any().any():
    first_valid = df.dropna().index.min()
    df.loc[: first_valid + pd.Timedelta(days=3)] = (
        df.loc[: first_valid + pd.Timedelta(days=3)].ffill()
)
```

I.2 — Indicateurs « sans look-ahead » & masques bruts

Ex. Bandes de Bollinger (20, 2σ) en contrarian :

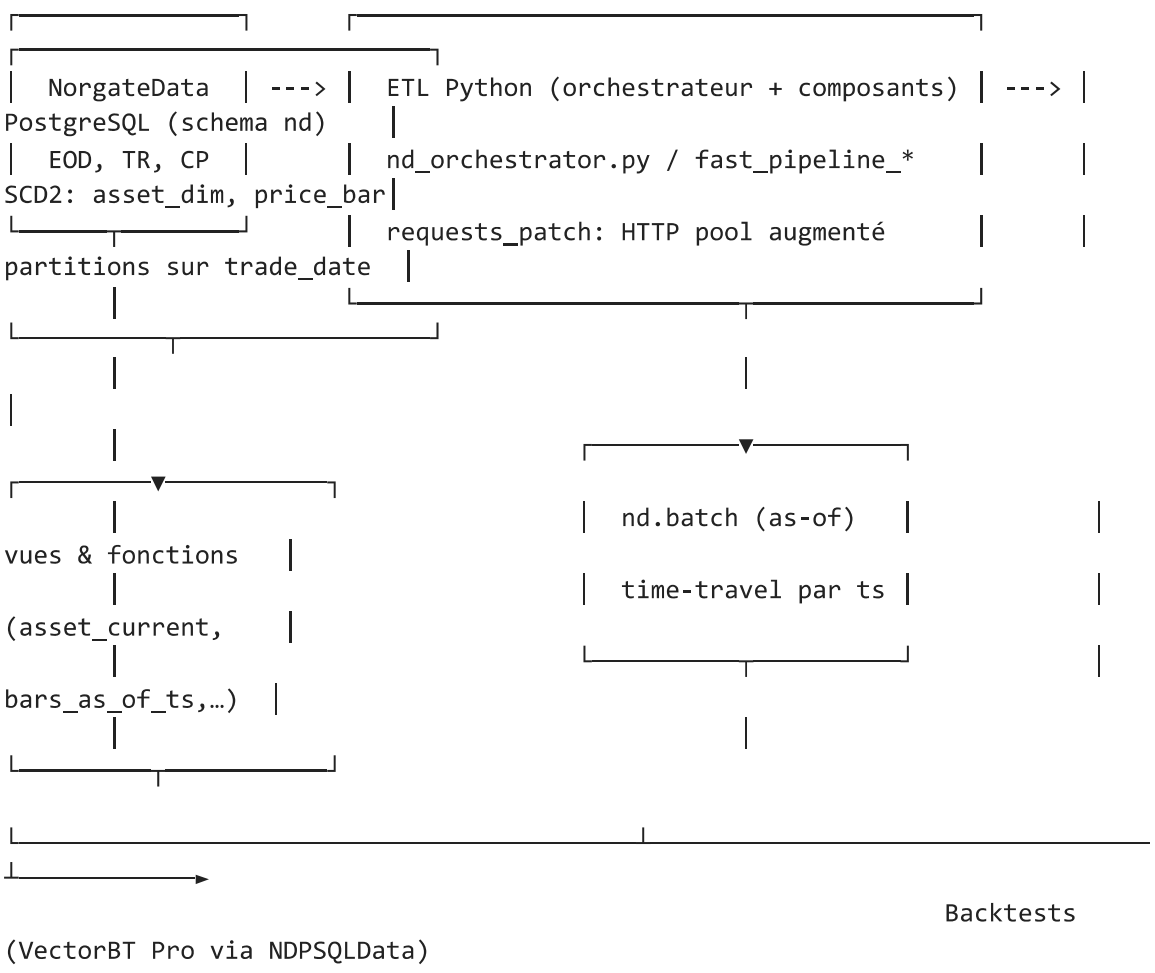
Annexe B — ETL NorgateData (v5) : Architecture, Schéma `nd`, Exécution

Cette annexe reflète la version v5 de l'ETL : orchestrateur multiprocess, drivers d'exécution (full/incr/sync), schéma SQL `nd` (SCD2 complet) avec fonctions "as-of", règles DQ, commandes types, intégration VectorBT, et opérations.

B.1 — Architecture & composants

Flux : NorgateData (EOD • Total Return • Current & Past) → ETL Python (multiprocess CPU + multithread IO) → PostgreSQL — schéma `nd` (SCD2, partitions par année) → Snapshots/batch (time-travel) → Moteur de simulation (VectorBT Pro).

Schéma B.1 — Orchestrateur & schéma `nd`



Composants clés :

• `etl/nd_orchestrator.py` — superviseur multiprocess (budget global, backpressure, autoscaling writers).

Annexe C — Inventaire des sources & scripts (v5)

Inventaire des fichiers fournis dans le ZIP d'annexes (version v5) avec leur rôle, leurs entrées/sorties et les commandes/options clés. Les chemins sont **relatifs à la racine du ZIP**.

Fichier (ZIP → chemin)	Rôle	Entrées / Sorties	Commande / Options clés
03_run_full.py	Full historique (watchlist/whitelist)	In : NorgateData • Out : `nd.asset_dim`/ `nd.price_bar`/ `nd.batch`	`python 03_run_full.py --interval D --start-date 1900-01-01 --watchlist MaWatchlist`
03_run_incr.py	Incrémental (**lookback paramétrable**)	In : NorgateData • Out : versions SCD2	`python 03_run_incr.py --interval D --lookback-days 7`
04_sync_metadata.py	Sync référentiel instruments (**SCD2 `nd.asset_dim`**)	In : NorgateData • Out : `nd.asset_dim`	`python 04_sync_metadata.py`
05_sync_fundamentals.py	Sync fondamentaux (si souscrits)	In : NorgateData • Out : tables fundamentals	`python 05_sync_fundamentals.py`
06_sync_index_memberships.py	Sync appartenance indices	In : NorgateData • Out : tables d'index	`python 06_sync_index_memberships.py` *(utilise `06_nd_index_membership.sql`)*

- Gray, W. R., & Vogel, J. R. (2016). *Quantitative Momentum: A Practitioner's Guide to Building a Momentum-Based Stock Selection System*.

- Carlisle, T. E. (2017). *The Acquirer's Multiple: How the Billionaire Contrarians of Deep Value Beat the Market*.

- VectorBT® PRO. (2025). https://vectorbt.pro

Les fichiers TP_VectorBT_Pro sont disponibles via le lien OneDrive suivant : https://groupeesaip-my.sharepoint.com/:f:/g/personal/mrousseau_ing2027_esaip_org/EhK9mDCv68pPknk5X8z2iDkBC66kBVP4_Y9c_zs0jhS33Q?e=nDHhwJ