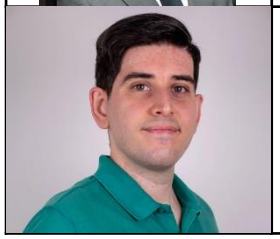
BIG DATA MANAGEMENT

Hello!

Team members for group project:

	•••	D’Alessandro Giammarco	1753102
	•••	Ionta Antonio	1469982

Abstract

Analyze data about football players coming from heterogeneous sources.

- ▷ Information Integration according with Data Warehousing principles.
- ▷ Materialization of integration results.
- ▷ Loading into Data Lake.
- ▷ Visual Analytics for Business Intelligence.

1.

The Data

Soccer Players Performances

Source Datasets

FBref

It contains statistics about real soccer player performances for the Big 5 European leagues, for seasons from 2012/2013 to 2021/2022.

Data comes from FBref.com, famous international Web portal devoted to tracking statistics for football teams and players from around the world.

FM20

It contains fictional player attributes contained in the database of Football Manager 2020 (season 2019/20) videogame.

Football Manager simulation gaming has become particularly interesting also in real soccer scouting activities for its realistic recreation of professional world of football.

https://www.footballmanager.com/features/data-hub

The source datasets come from two distinct kaggle repositories.

They both deals with data about football players, but from two completely different perspectives.

Datasets are heterogeneous in schema and semantics, but not in data format, which is in both cases CSV.

13 CSV files

12 for FBref dataset + 1 for FM dataset

346,797 records

202,047 for FBref dataset + 144,750 for FM dataset

395 attributes

332 for FBref dataset + 63 for FM dataset

?

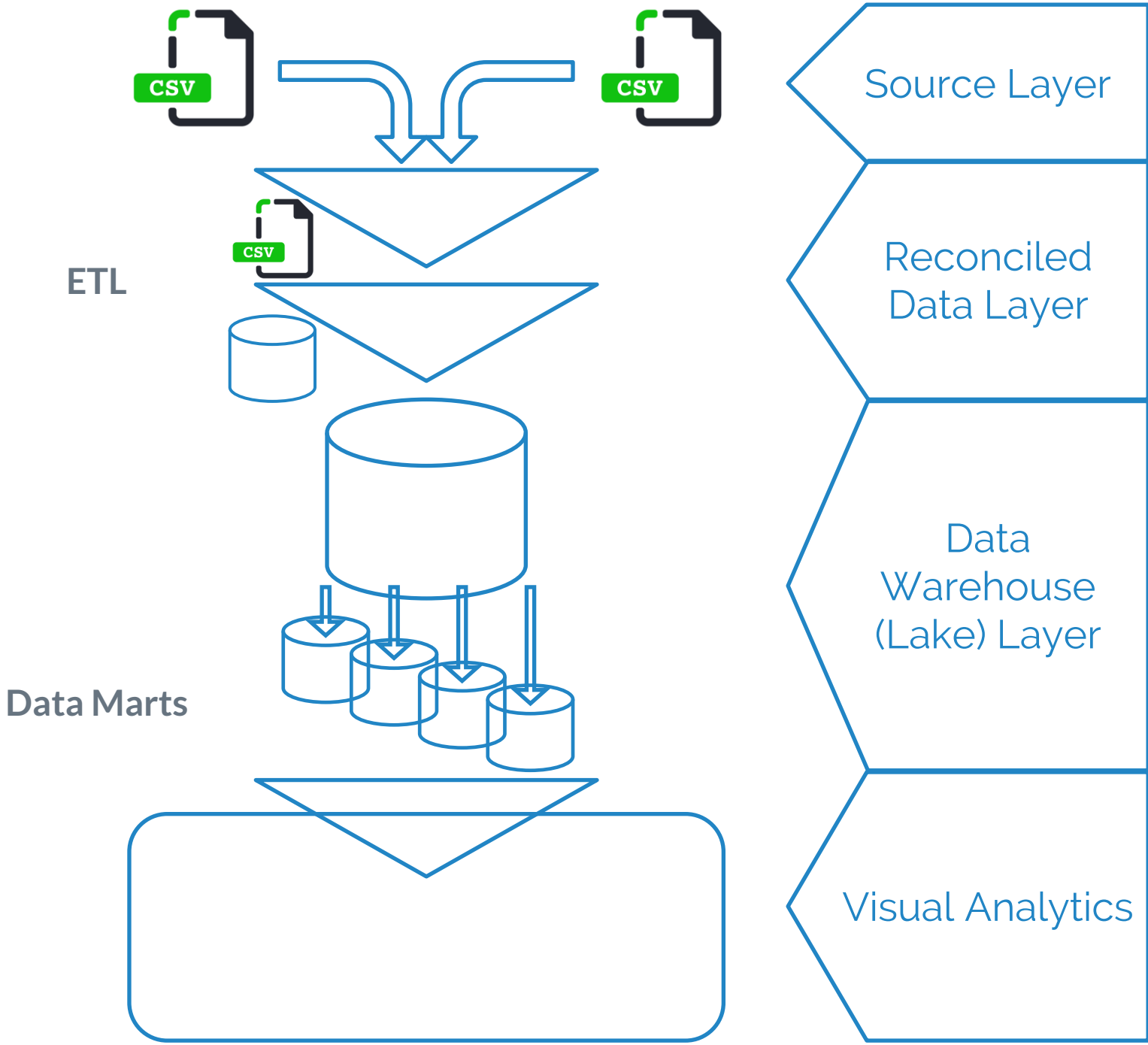
Is there a sort of relationship between hypothetic player attributes and actual performance on the field?

2.

Data Warehose Design

From Conceptual Modeling to Logical Realization

Three-Layer Architecture

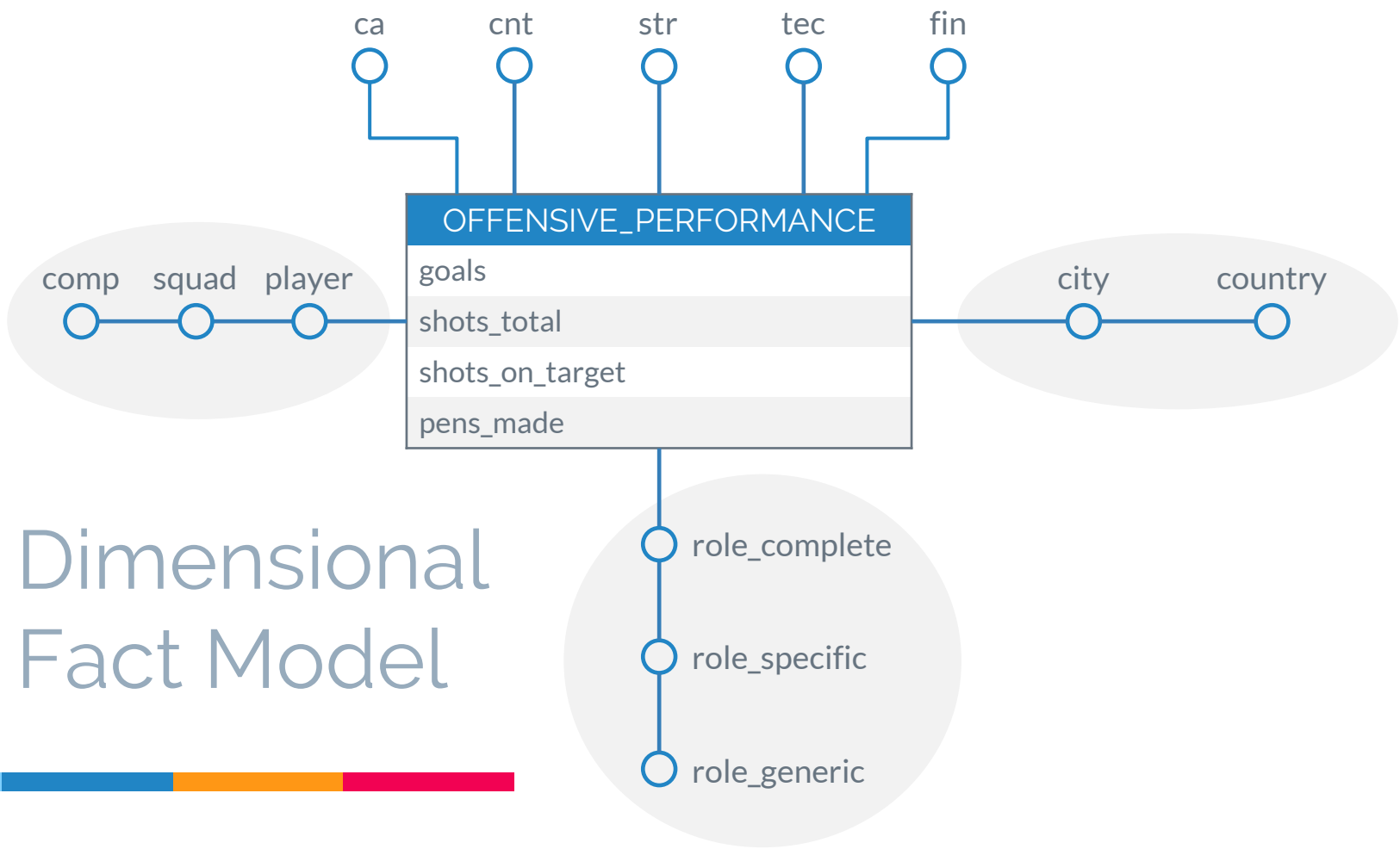


full source schema here

Source Schema

Defense/31	Contains FBref data on a player's defensive performance.
GCA/24	Contains FBref data on a player's goal and shot creation.
Info/13	Contains FBref general information about a player.
Keeper/26	Contains FBref data on goalkeeper performance.
KeeperAdv/33	Contains FBref advanced data for goalkeeper performance.
Misc/24	Contains FBref miscellaneous player performance data.
Passing/30	Contains FBref passing data.
PassingTypes/33	Contains FBref data about players' pass types.
PlayingTime/29	Contains FBref data about a player's playing time.
Possession/32	Contains FBref possession data.
Shooting/25	Contains FBref data about a player's shooting performance.
Standard/32	Contains an overview of FBref player performance data.
FM20/64	Contains Football Manager 2020 videogame fictional player attributes.

full DFM here



Dimensional Fact Model

full global schema here

Global Relational Schema

GoalkeeperPerformance$_{/8}$	FACT	Contains players real statistics meaningful for summarizing their performance as goalkeepers.
DefensivePerformance$_{/8}$	FACT	Contains players real statistics meaningful for summarizing their defensive performance.
PlaymakingPerformance$_{/12}$	FACT	Contains players real statistics meaningful for summarizing their performance playmaking-related.
OffensivePerformance$_{/8}$	FACT	Contains players real statistics meaningful for summarizing their offensive performance.
Birth$_{/3}$	DIMENSION	Contains players' cities and countries of birth.
Role$_{/4}$	DIMENSION	Contains information about the roles assumed by players on pitch with different levels of precision.
PlayerOrganization$_{/4}$	DIMENSION	Contains players full names and the squads and the national competitions to which they belong.

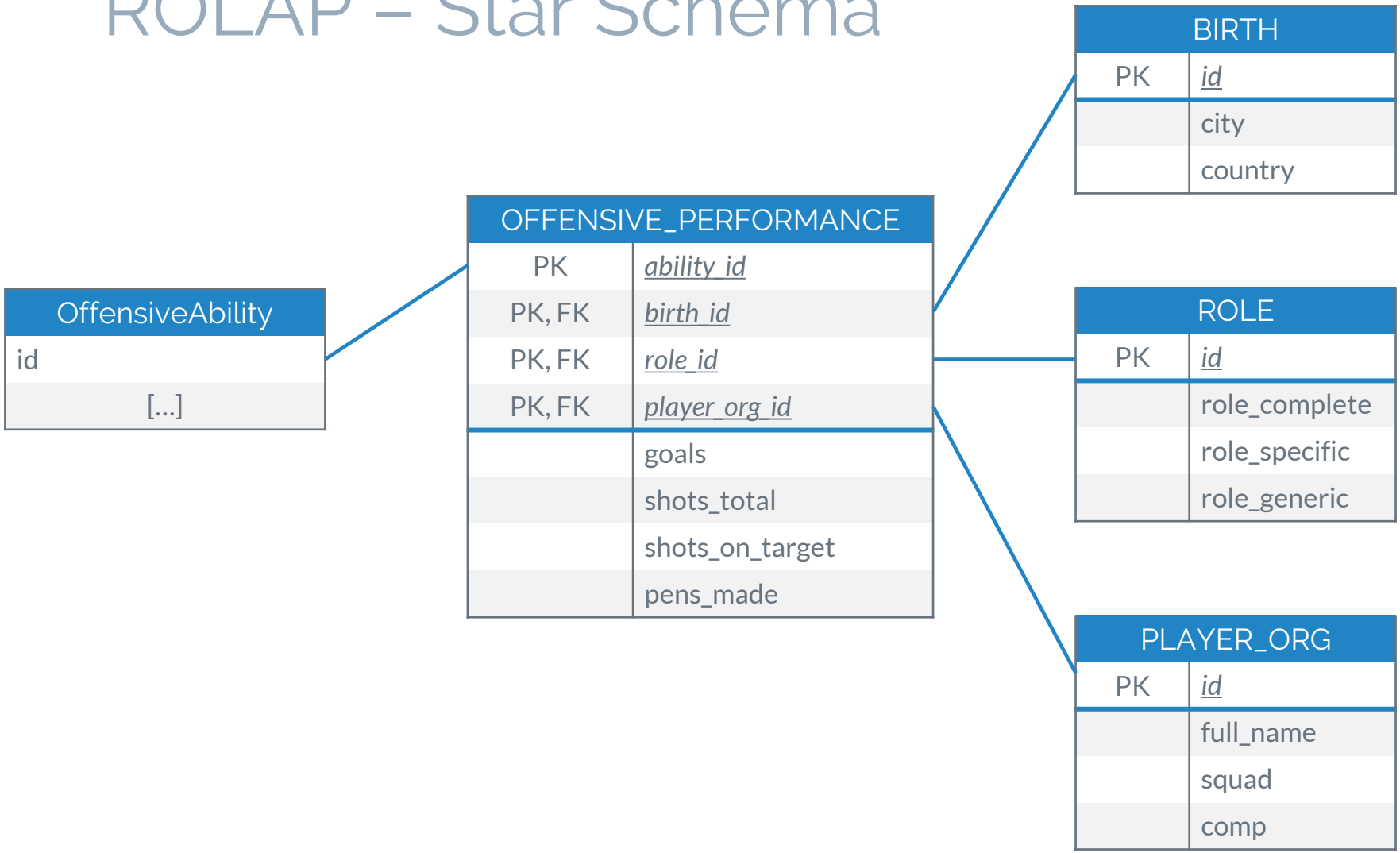
full global schema here

Global Non-Relational Structure

GoalkeeperAbility/6	DIMENSION	Contains most relevant fictional attributes related to players' ability as goalkeepers.
DefensiveAbility/6	DIMENSION	Contains most relevant fictional attributes related to players' defensive ability.
PlaymakingAbility/8	DIMENSION	Contains most relevant fictional attributes related to players' ability playmaking-related.
OffensiveAbility/6	DIMENSION	Contains most relevant fictional attributes related to players' offensive ability.

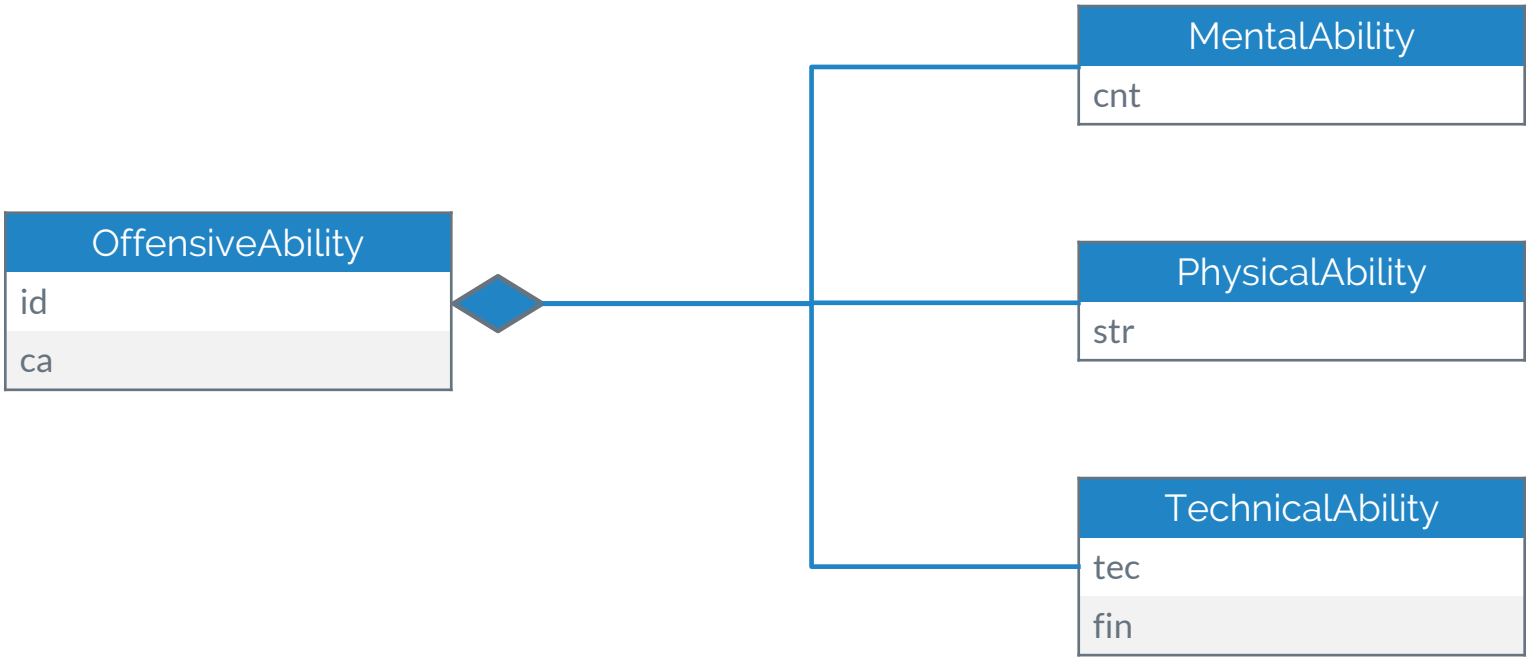
full star schema here

ROLAP – Star Schema



full star schema here

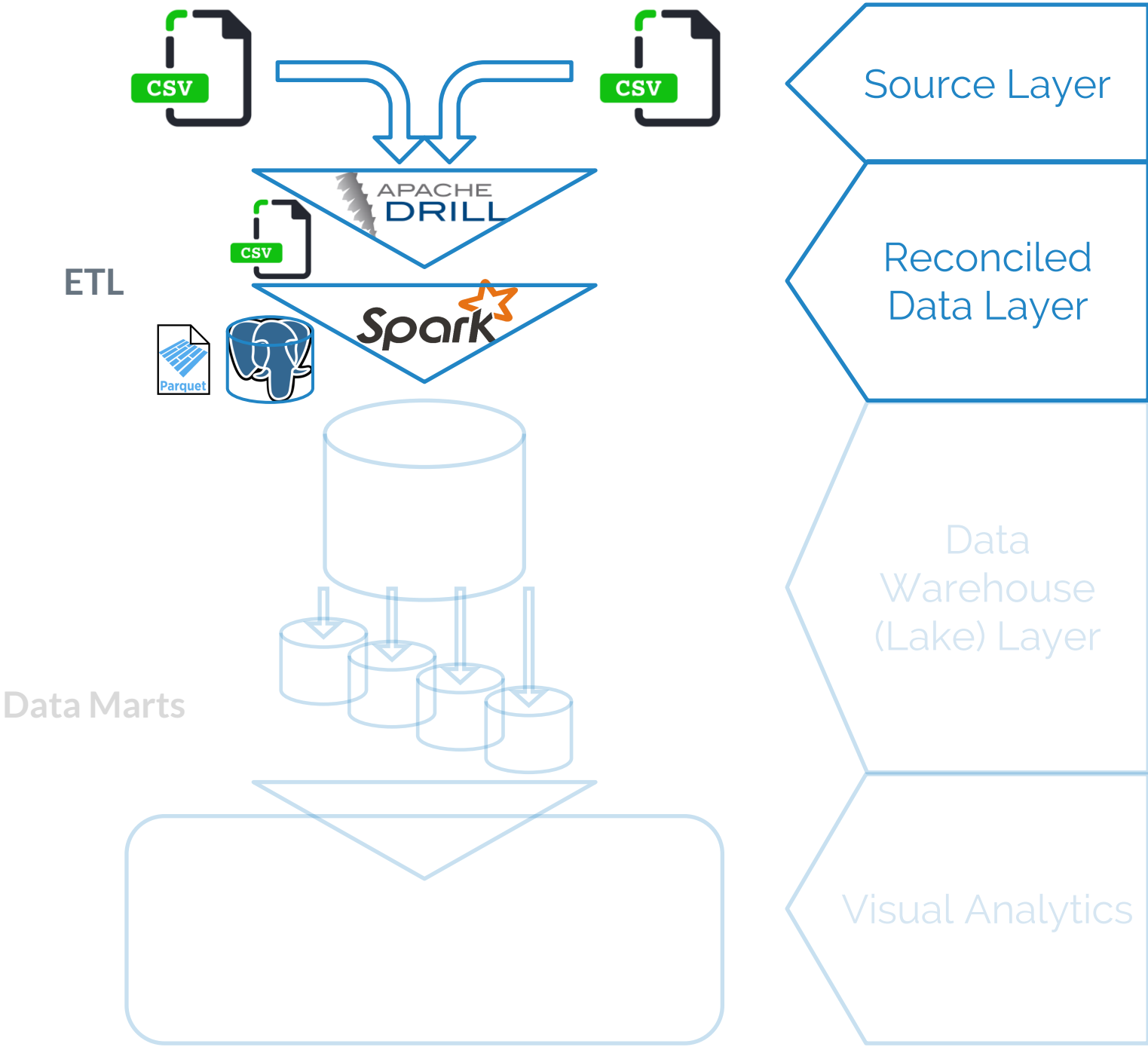
Aggregate Data Model



3.
Data Warehose Implementation

ETL

Three-Layer Architecture



Extraction | Transformation | Loading

Source

- 13 CSV files
- 346,797 records
- 395 attributes

APACHE DRILL

SQL engine designed to enable data exploration and analytics on non-relational datastores.

Staging Area

- 11 CSV files
- 16,868 records
- 229 attributes

Extraction | Transformation | Loading

Source

- 13 CSV files
- 346,797 records
- 395 attributes

APACHE DRILL

SQL engine designed to enable data exploration and analytics on non-relational datastores.

Staging Area

- 11 CSV files
- 16,868 records
- 229 attributes

```
alter session set `store.format`='csv';

create table export.fbrefStandardExtract as
select
id,season,country,comp_level,lg_finish,squad,age,games,games_starts,mnts,minutes_90s,goals,assists,goals_pens,pens_made,pens_att,cards_yellow,cards_red,goals_per90,assists_per90,goals_assists_per90,goals_pens_per90
from input.`fbref-standard.csv`
Where season = '2019-2020'
    and country in ('ITA', 'GER', 'ESP', 'ENG')
    and comp_level in ('1. Serie A', '1. Bundesliga', '1. La Liga', '1. Premier League');
```

Extraction | Transformation | Loading

Source

- 13 CSV files
- 346,797 records
- 395 attributes

APACHE DRILL

SQL engine designed to enable data exploration and analytics on non-relational datastores.

Staging Area

- 11 CSV files
- 16,868 records
- 229 attributes

Scala Spark

Multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

Slick

Modern database query and access library for Scala. It features an extensible query compiler which can generate code for different backends.

Reconciled Data

- 7 tables
- 4 parquet files
- 8,101 records
- 73 attributes

Scala-Slick
Relational Model

```
class OffensivePerformanceEntity(tag: Tag) extends Table[OffensivePerformance](tag, "OFFENSIVE_PERFORMANCE") {
    def offensiveAbilityId = column[Option[Long]]("offensive_ability_id");
    def birthId = column[Option[Long]]("birth_id");
    def birth = foreignKey("birth_fk", birthId, TableQuery[BirthEntity])(_.id, onDelete =
ForeignKeyAction.Cascade)
    def roleId = column[Option[Long]]("role_id");
    def role = foreignKey("role_fk", roleId, TableQuery[RoleEntity])(_.id, onDelete = ForeignKeyAction.Cascade)
    def playerOrganizationId = column[Option[Long]]("player_organization_id");
    def playerOrganization = foreignKey("player_organization_fk", playerOrganizationId,
TableQuery[PlayerOrganizationEntity])(_.id)

    def pk = primaryKey("offensive_performance_pk", (offensiveAbilityId, birthId, roleId, playerOrganizationId))

    def goals = column[Option[Int]]("goals")
    def shotsTotal = column[Option[Int]]("shots_toal")
    def shotsOnTarget = column[Option[Int]]("shots_on_target")
    def pensMade = column[Option[Int]]("pens_made")

    def * = (offensiveAbilityId, birthId, roleId, playerOrganizationId, goals, shotsTotal, shotsOnTarget,
pensMade) <> (OffensivePerformance.tupled, OffensivePerformance.unapply)
}
```

Scala
Non-Relational Model

```
case class OffensiveMentalAbility(
    cnt: Int
)

case class OffensivePhysicalAbility(
    str: Int
)

case class OffensiveTechnicalAbility(
    tec: Int,
    fin: Int
)

case class OffensiveAbility(
    id: Long,
    ca: Int,
    mentalAbility: OffensiveMentalAbility,
    physicalAbility: OffensivePhysicalAbility,
    technicalAbility: OffensiveTechnicalAbility
)
```

Spark SQL OffensivePerformanceFull

```
select
    cast(sh.goals as integer),
    cast(sh.shots_total as integer) as shotsTotal,
    cast(sh.shots_on_target as integer) as shotsOnTarget,
    cast(pens_made as integer) as pensMade,
    info.id as fbrefId,
    info.name as playerFullName,
    cast(CA as integer) as ca,
    cast(Str as integer) as str,
    cast(Tec as integer) as tec,
    cast(Fin as integer) as fin,
    cast(Cnt as integer) as cnt,
    cityob as city,
    countryob as country,
    info.position as position,
    sh.squad,
    sh.comp_level as comp
from
    FbrefInfo info
    join FbrefShooting sh on info.id = sh.id
    join Fm20 on info.name = Fm20.Name
```

Scala+Spark OffensiveAbility

```
select distinct
    ca,
    cnt,
    str,
    tec,
    fin
from
    OffensivePerformanceFull
```

.withColumn("id", monotonically_increasing_id())

```
def offensive_ability_mapper(r: Row):
OffensiveAbility = {
     val id = r.getAs[Long]("id")
     val ca = r.getAs[Int]("ca")
     val cnt = r.getAs[Int]("cnt")
     val str = r.getAs[Int]("str")
     val tec = r.getAs[Int]("tec")
     val fin = r.getAs[Int]("fin")

     OffensiveAbility(
         id,
         ca,
         OffensiveMentalAbility(cnt),
         OffensivePhysicalAbility(str),
         OffensiveTechnicalAbility(tec, fin)
     )
}
```

Scala+Spark+Slick PlayerOrganization

```
select distinct
    playerFullName as player,
    squad,
    comp
from GoalkeeperPerformanceFull
union
select distinct
    playerFullName as player,
    squad,
    comp
from DefensivePerformanceFull
union
select distinct
    playerFullName as player,
    squad,
    comp
from PlaymakingPerformanceFull
union
select distinct
    playerFullName as player,
    squad,
    comp
from OffensivePerformanceFull
```

```
.withColumn("id",
monotonically_increasing_id())
```

```
val playerOrganizationTable =
TableQuery[PlayerOrganizationEntity]

val playerOrganizationList =
dataframe_to_list(playerOrganizationDf,
Encoders.product[PlayerOrganization])

[…]

playerOrganizationTable.schema.create

playerOrganizationTable ++=
playerOrganizationList
```

Scala+Spark+Slick OffensivePerformance

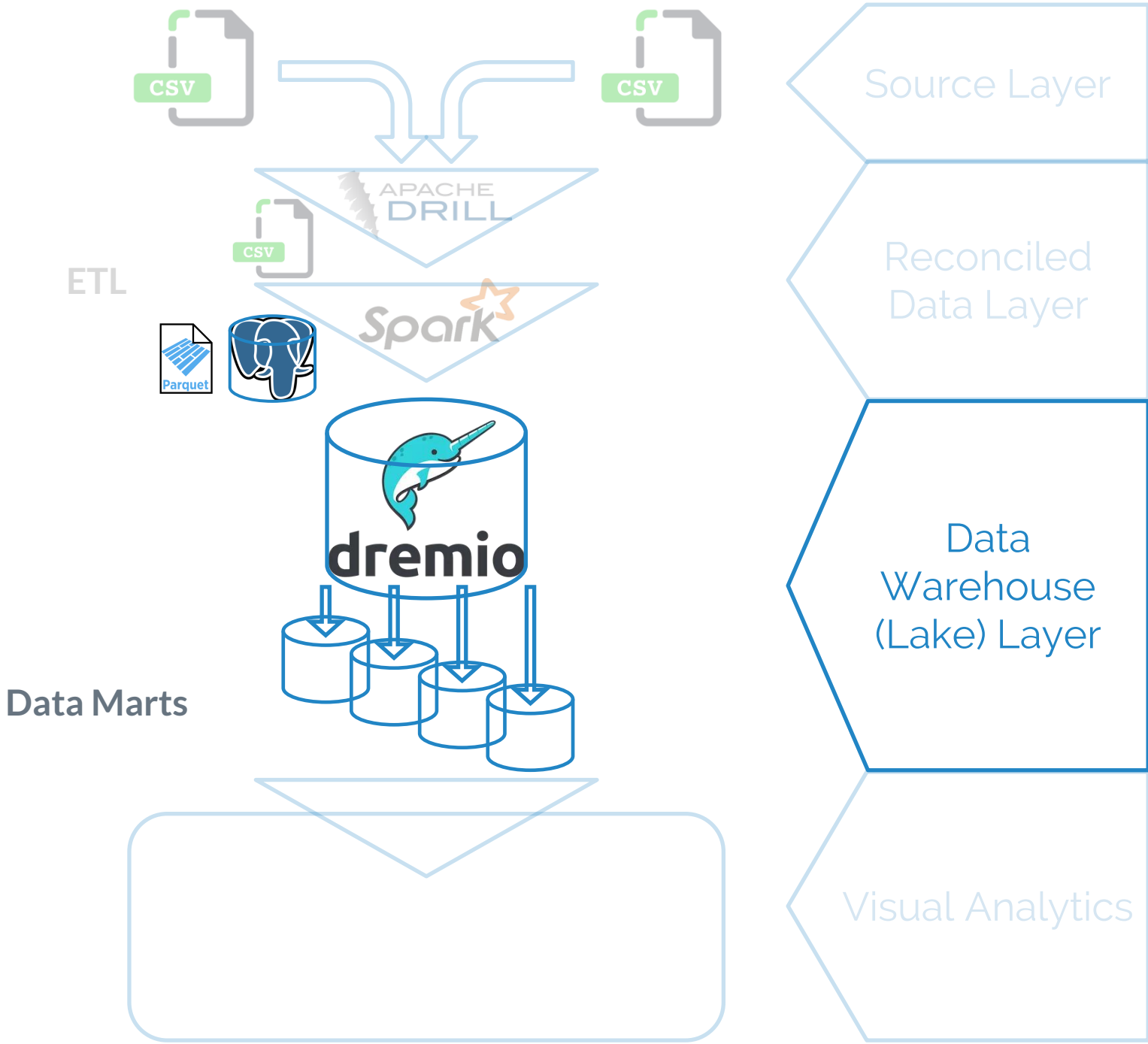
```
select
    oa.id as offensiveAbilityId,
    brt.id as birthId,
    rl.id as roleId,
    porg.id as playerOrganizationId,
    goals,
    shotsTotal,
    shotsOnTarget,
    pensMade
from
    OffensivePerformanceFull opf
    join OffensiveAbility oa on (opf.ca = oa.ca and
opf.cnt = oa.cnt and opf.str = oa.str and opf.tec =
oa.tec and opf.fin = oa.fin)
    join PlayerOrganization porg on (opf.playerFullName
= porg.player and opf.squad = porg.squad and opf.comp =
porg.comp)
    join Birth brt on (opf.city = brt.city and
opf.country = brt.country)
    join Role rl on (opf.position = rl.role_complete)
```

```
val offensivePerformanceTable =
TableQuery[OffensivePerformanceEntity]

val offensivePerformanceList =
dataframe_to_list(offensivePerformanceDf,
Encoders.product[OffensivePerformance])

[…]

offensivePerformanceTable.schema.create

offensivePerformanceTable ++=
offensivePerformanceList
```

4.
Flow into Data Lakehouse

OLAP with Dremio

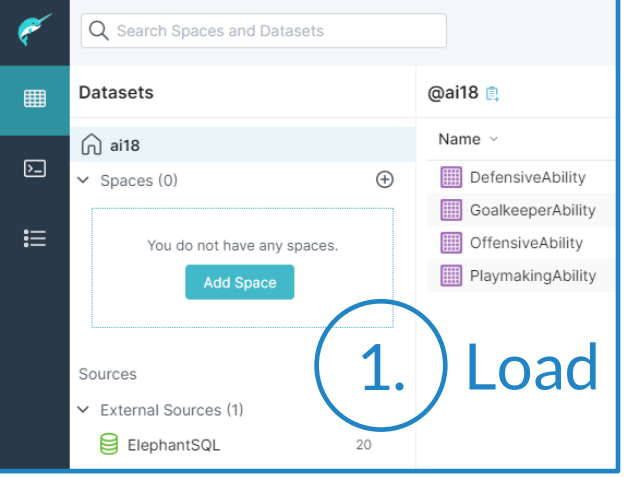
Three-Layer Architecture

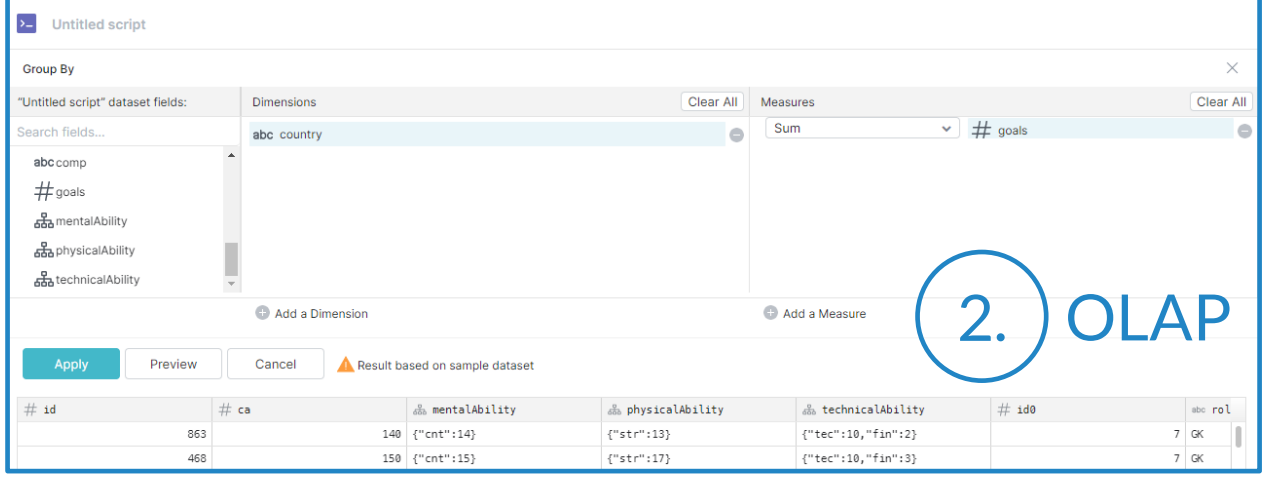


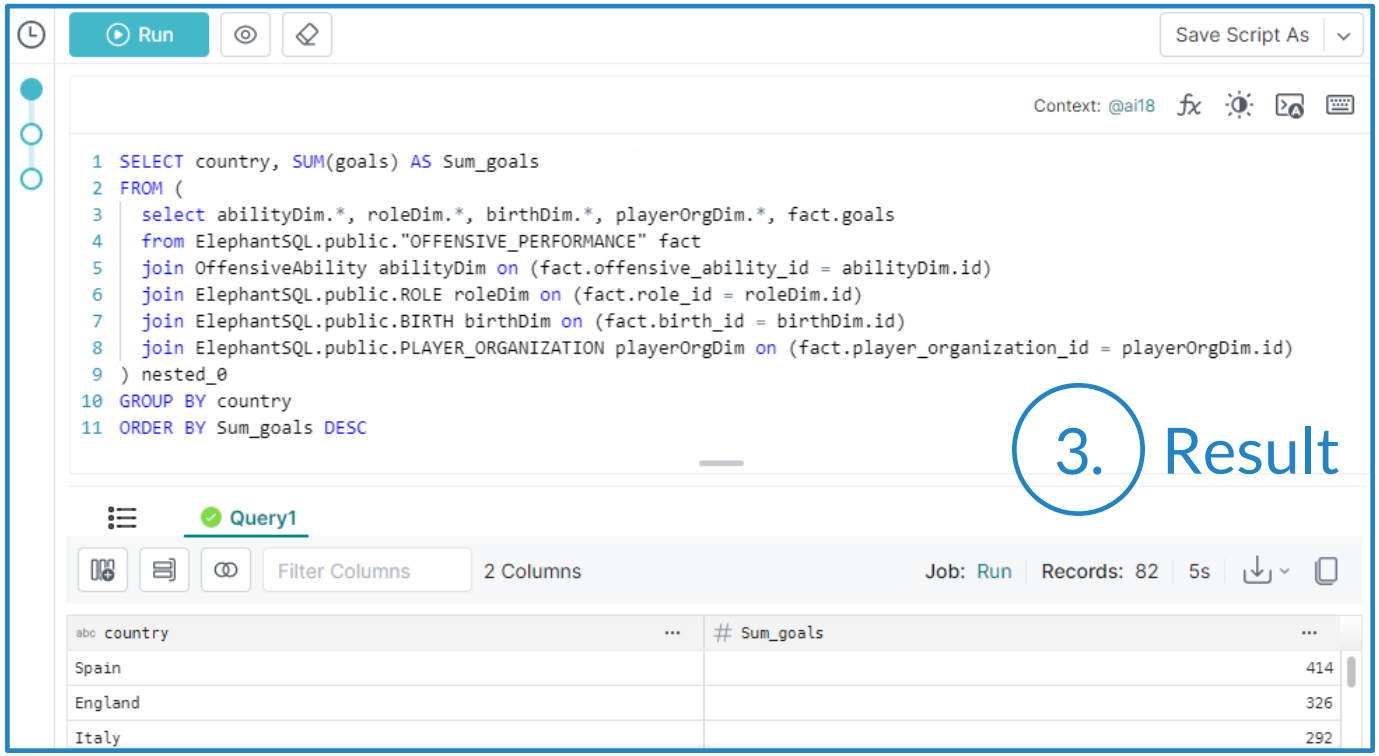
dremio

The Easy and Open Data Lakehouse

- Self-service analytics with data warehouse functionality and data lake flexibility across all of your data.
- Transactional support (ACID).
- Data quality and governance.
- Support for schema management.


Search Spaces and Datasets
Datasets
@ai18
ai18
Spaces (0)
You do not have any spaces.
Add Space
Name
DefensiveAbility
GoalkeeperAbility
OffensiveAbility
PlaymakingAbility
Sources
External Sources (1)
ElephantSQL
20
1.
Load

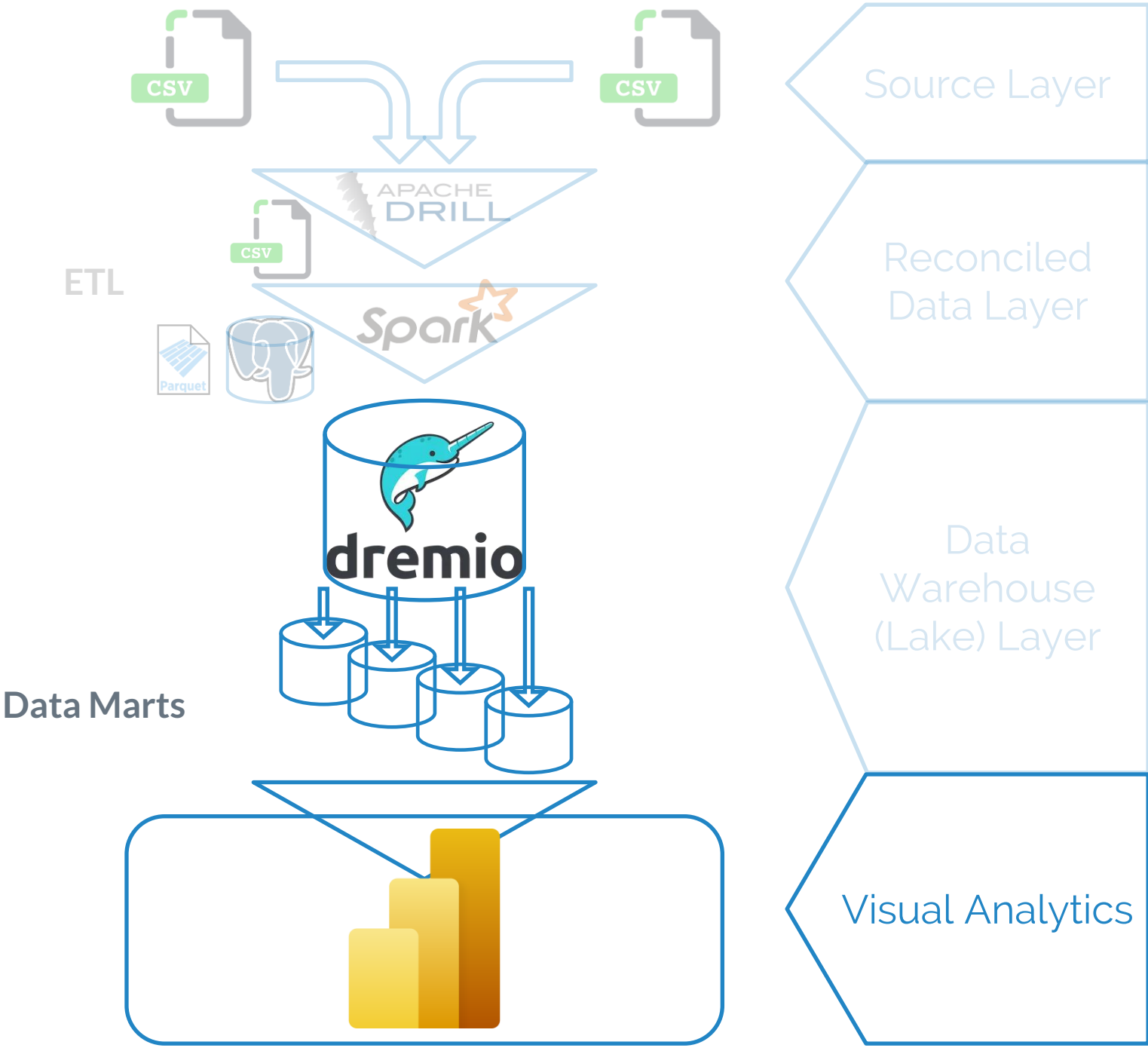

Untitled script
Group By
"Untitled script" dataset fields:
Dimensions
Clear All
Measures
Clear All
Search fields...
abc comp
goals
mentalAbility
physicalAbility
technicalAbility
abc country
Sum
goals
Add a Dimension
Add a Measure
Apply
Preview
Cancel
Result based on sample dataset
id
ca
mentalAbility
physicalAbility
technicalAbility
id0
rol
863
140
{"cnt":14}
{"str":13}
{"tec":10,"fin":2}
7
GK
468
150
{"cnt":15}
{"str":17}
{"tec":10,"fin":3}
7
GK
2.
OLAP


Run
Save Script As
Context: @ai18
SELECT country, SUM(goals) AS Sum_goals
FROM (
select abilityDim.*, roleDim.*, birthDim.*, playerOrgDim.*, fact.goals
from ElephantSQL.public."OFFENSIVE_PERFORMANCE" fact
join OffensiveAbility abilityDim on (fact.offensive_ability_id = abilityDim.id)
join ElephantSQL.public.ROLE roleDim on (fact.role_id = roleDim.id)
join ElephantSQL.public.BIRTH birthDim on (fact.birth_id = birthDim.id)
join ElephantSQL.public.PLAYER_ORGANIZATION playerOrgDim on (fact.player_organization_id = playerOrgDim.id)
) nested_0
GROUP BY country
ORDER BY Sum_goals DESC
3.
Result
Query1
Filter Columns
2 Columns
Job: Run
Records: 82
5s
country
Sum_goals
Spain
414
England
326
Italy
292

5.
Visual Analytics

Dashboard for Business Intelligence

Three-Layer Architecture

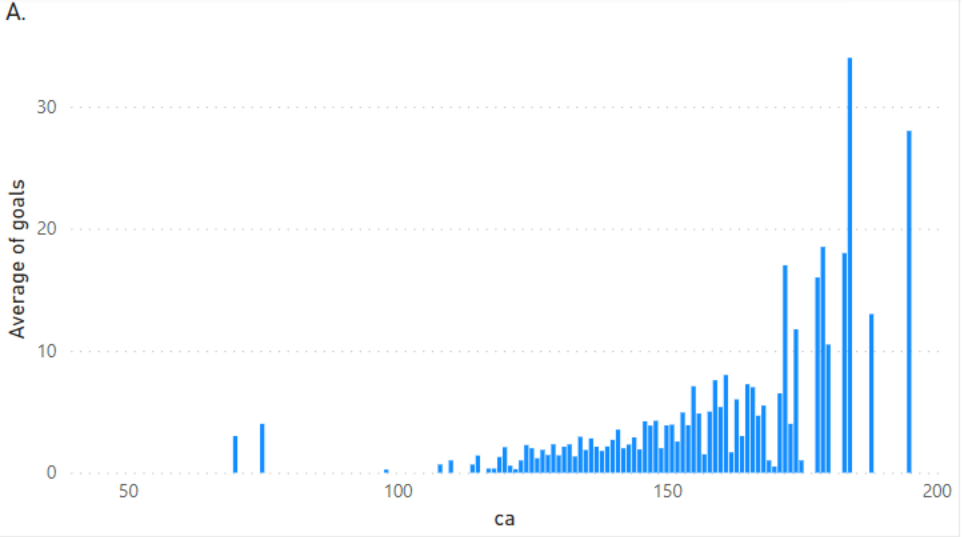


Microsoft Power BI

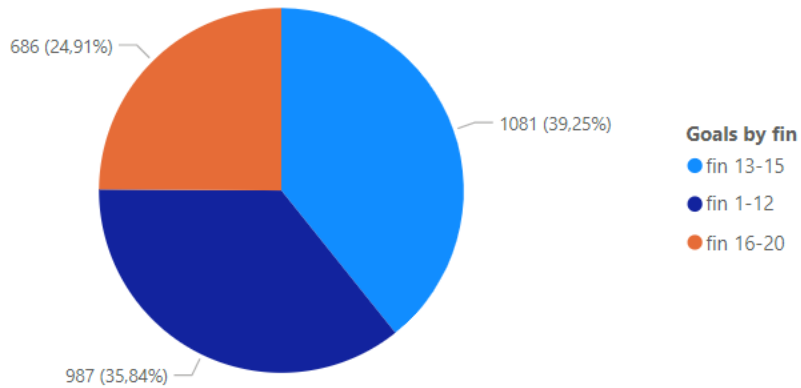
Business Intelligence platform for connecting to and visualizing heterogeneous data.

- Watch, read, and discover ways to unify, visualize, and securely share your data.
- Create data experiences which help you gain deeper data insight.
- Simplify big data analysis.
- Create data-centric culture.

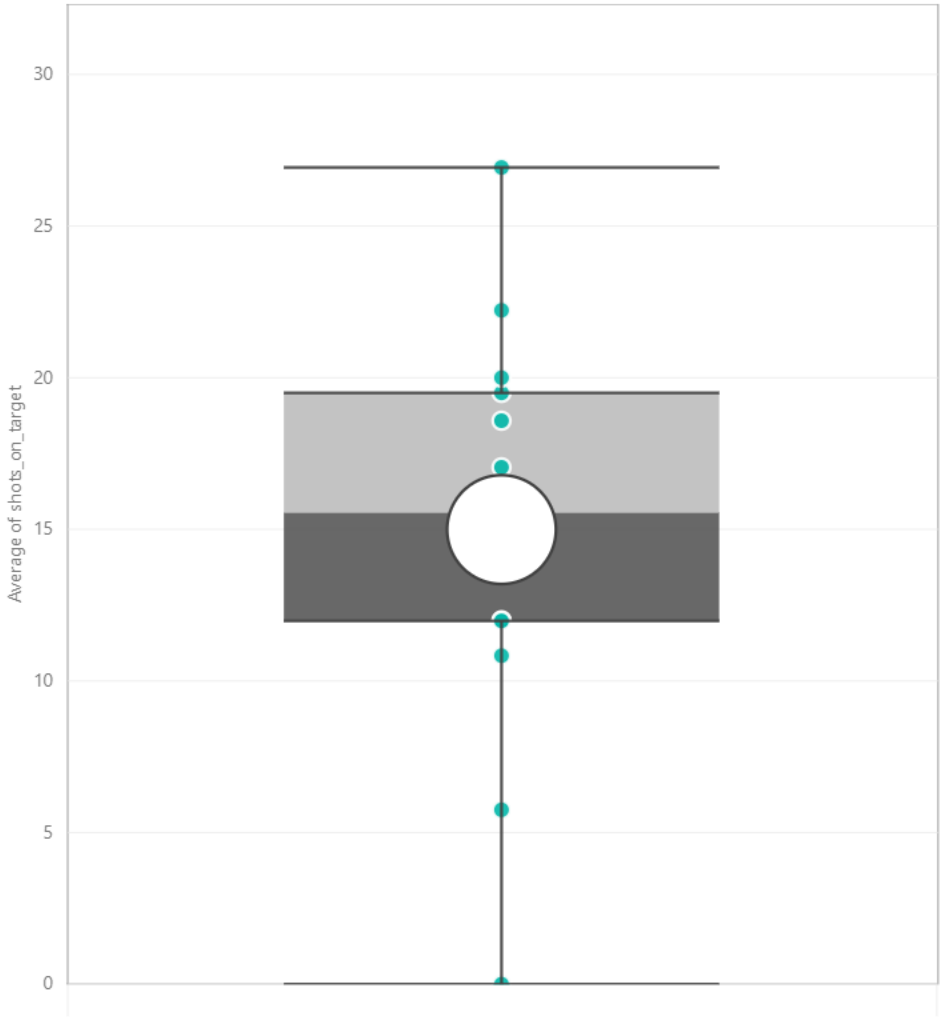
A.



B.



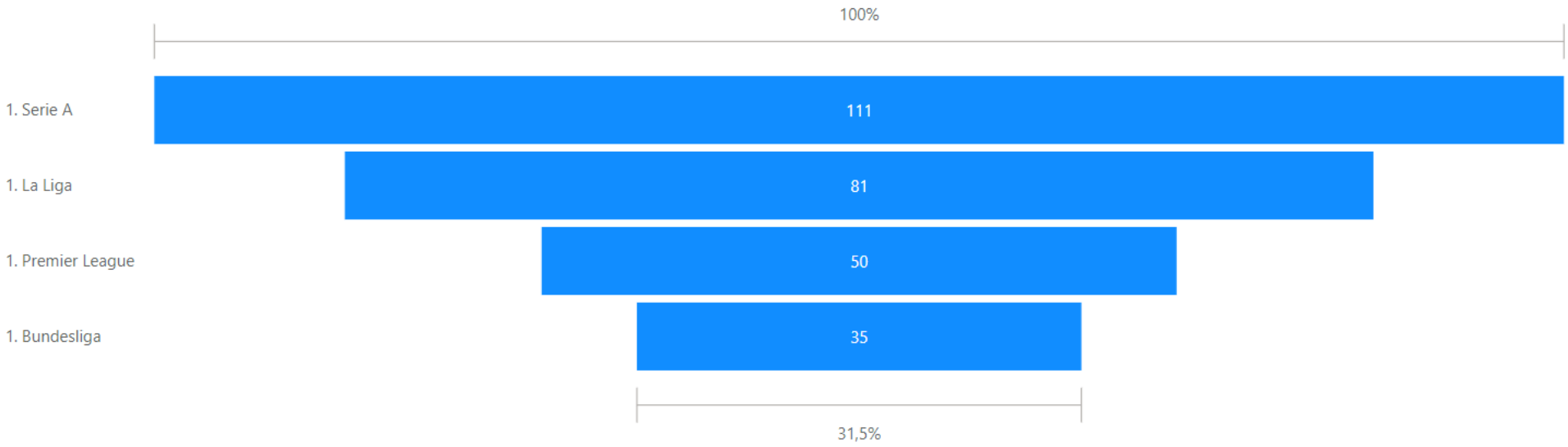
C.



Offensive Performance

Roll-up on ability dimension

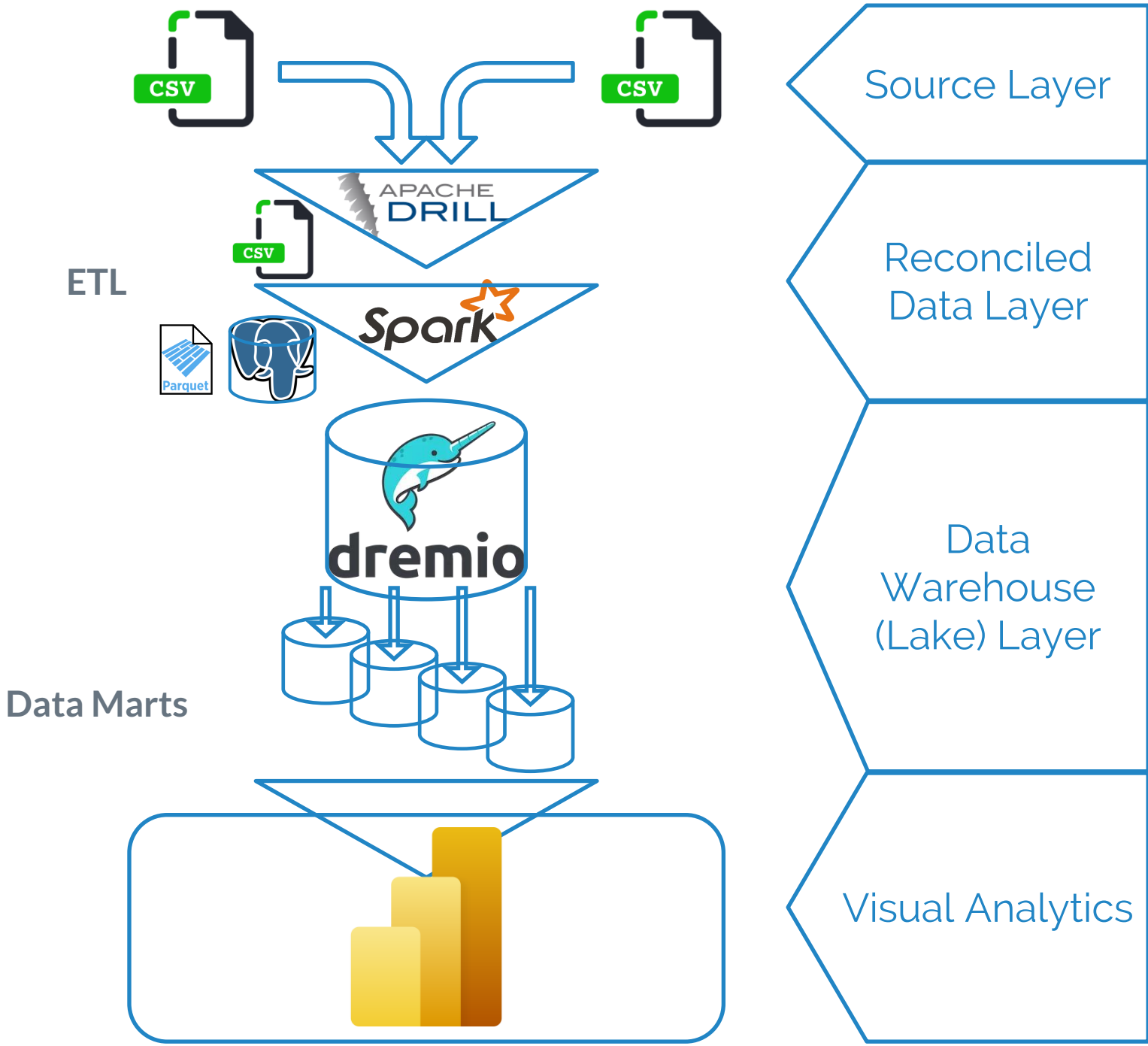
A.



Offensive Performance

Roll-up on organization dimension

The Full Picture



Thanks!

Share ideas and get involved!

Visit the GitHub page of the project