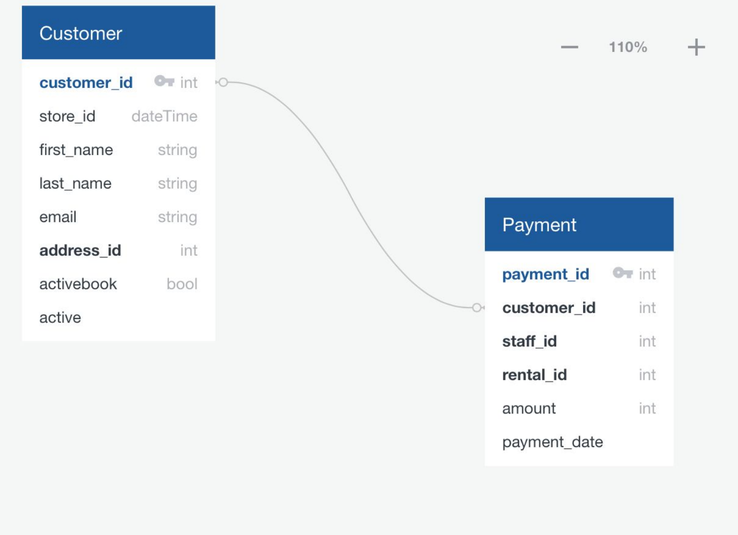
SQL

Exercises

Exercise #1 - Top 10 Customers

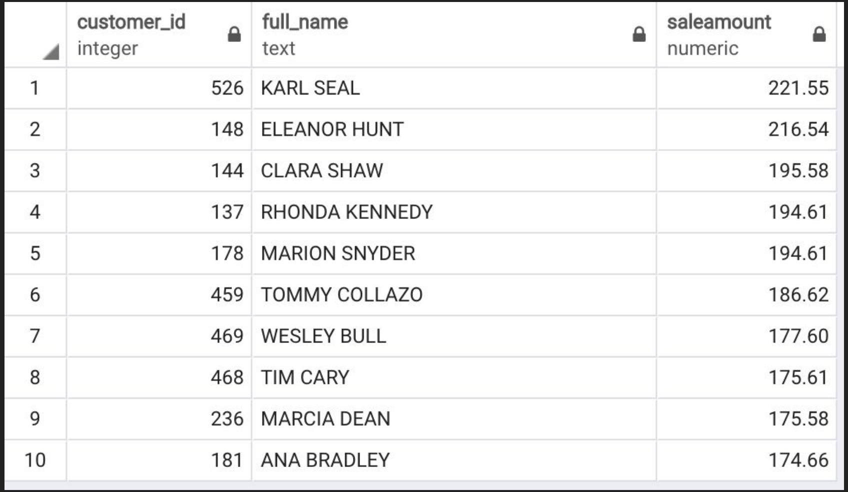


Customer

customer_id	int
store_id	dateTime
first_name	string
last_name	string
email	string
address_id	int
activebook	bool
active	

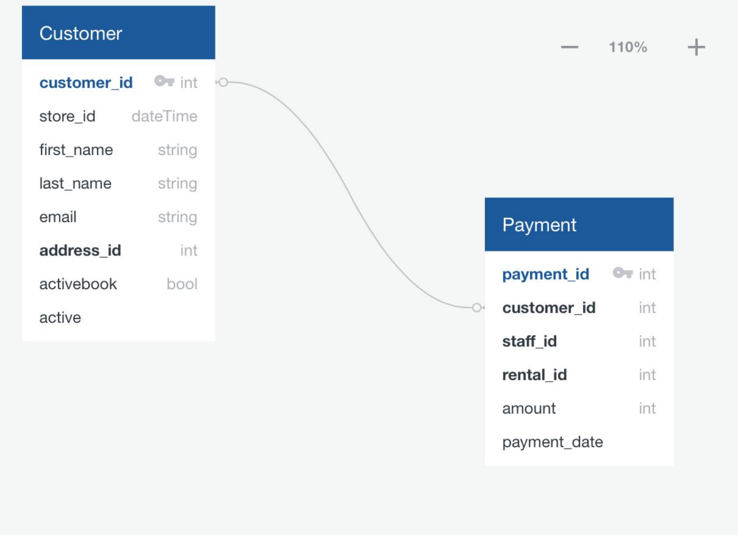
Payment

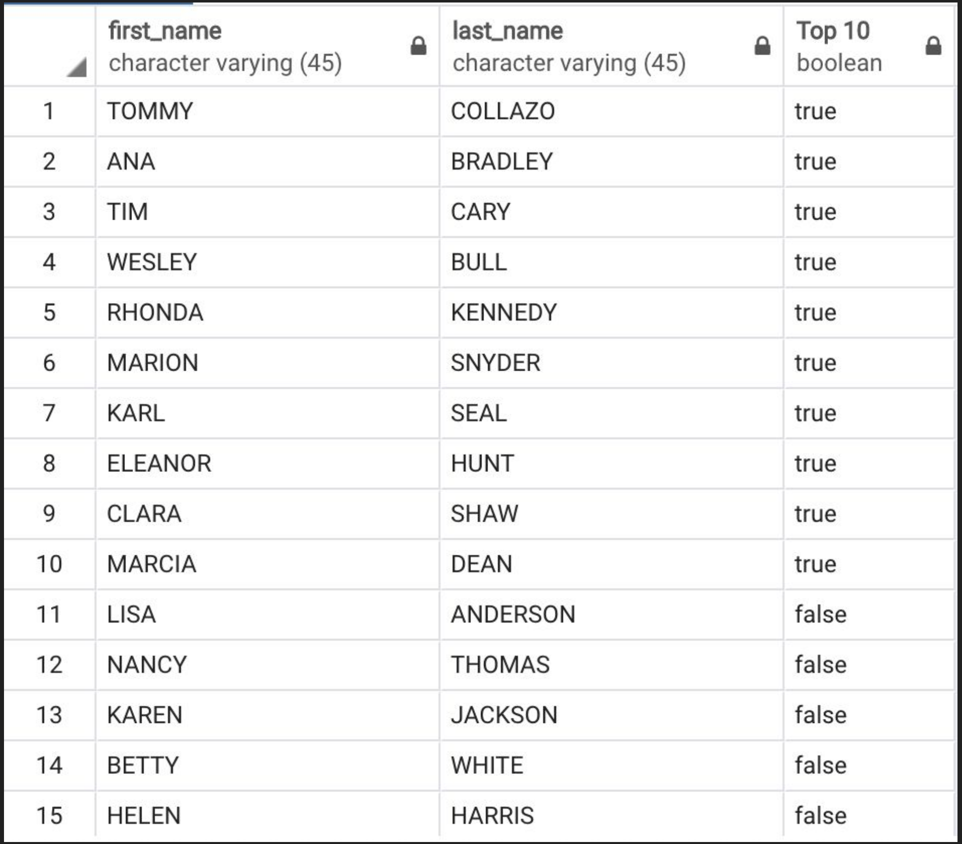
payment_id	int
customer_id	int
staff_id	int
rental_id	int
amount	int
payment_date	



	customer_id integer	full_name text	saleamount numeric
1	526	KARL SEAL	221.55
2	148	ELEANOR HUNT	216.54
3	144	CLARA SHAW	195.58
4	137	RHONDA KENNEDY	194.61
5	178	MARION SNYDER	194.61
6	459	TOMMY COLLAZO	186.62
7	469	WESLEY BULL	177.60
8	468	TIM CARY	175.61
9	236	MARCIA DEAN	175.58
10	181	ANA BRADLEY	174.66

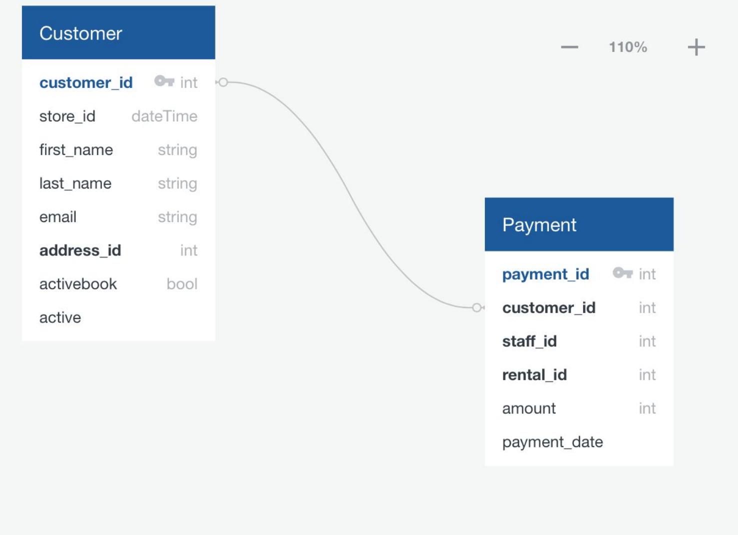
Exercise #2 - Show me all customers as well





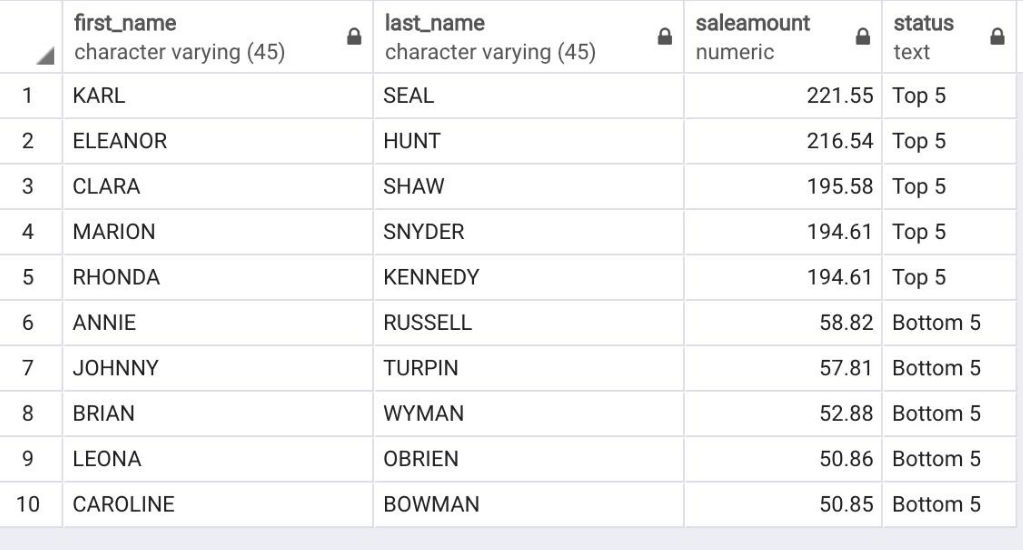
	first_name character varying (45)	last_name character varying (45)	Top 10 boolean
1	TOMMY	COLLAZO	true
2	ANA	BRADLEY	true
3	TIM	CARY	true
4	WESLEY	BULL	true
5	RHONDA	KENNEDY	true
6	MARION	SNYDER	true
7	KARL	SEAL	true
8	ELEANOR	HUNT	true
9	CLARA	SHAW	true
10	MARCIA	DEAN	true
11	LISA	ANDERSON	false
12	NANCY	THOMAS	false
13	KAREN	JACKSON	false
14	BETTY	WHITE	false
15	HELEN	HARRIS	false

Exercise #3 - Top 5 and Bottom 5



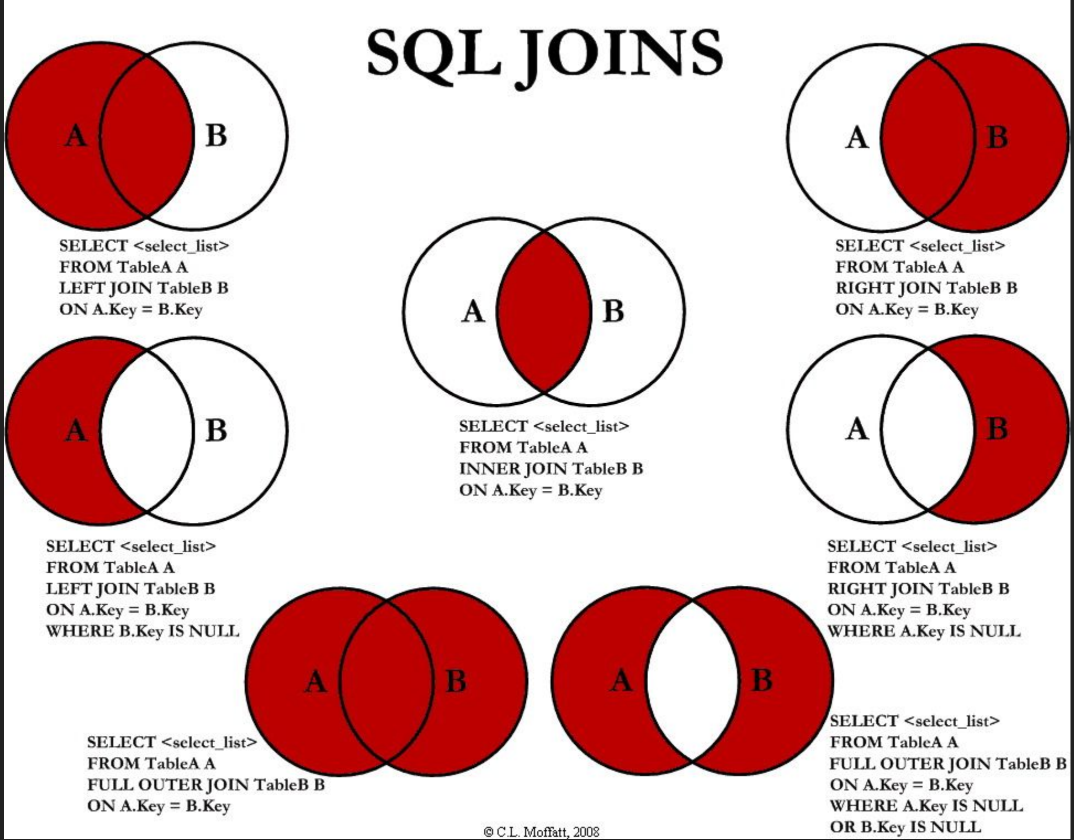
Customer	
customer_id	int
store_id	dateTime
first_name	string
last_name	string
email	string
address_id	int
activebook	bool
active	

Payment	
payment_id	int
customer_id	int
staff_id	int
rental_id	int
amount	int
payment_date	



	first_name character varying (45)	last_name character varying (45)	saleamount numeric	status text
1	KARL	SEAL	221.55	Top 5
2	ELEANOR	HUNT	216.54	Top 5
3	CLARA	SHAW	195.58	Top 5
4	MARION	SNYDER	194.61	Top 5
5	RHONDA	KENNEDY	194.61	Top 5
6	ANNIE	RUSSELL	58.82	Bottom 5
7	JOHNNY	TURPIN	57.81	Bottom 5
8	BRIAN	WYMAN	52.88	Bottom 5
9	LEONA	OBRIEN	50.86	Bottom 5
10	CAROLINE	BOWMAN	50.85	Bottom 5

SQL Joins - recap



SQL JOINS

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

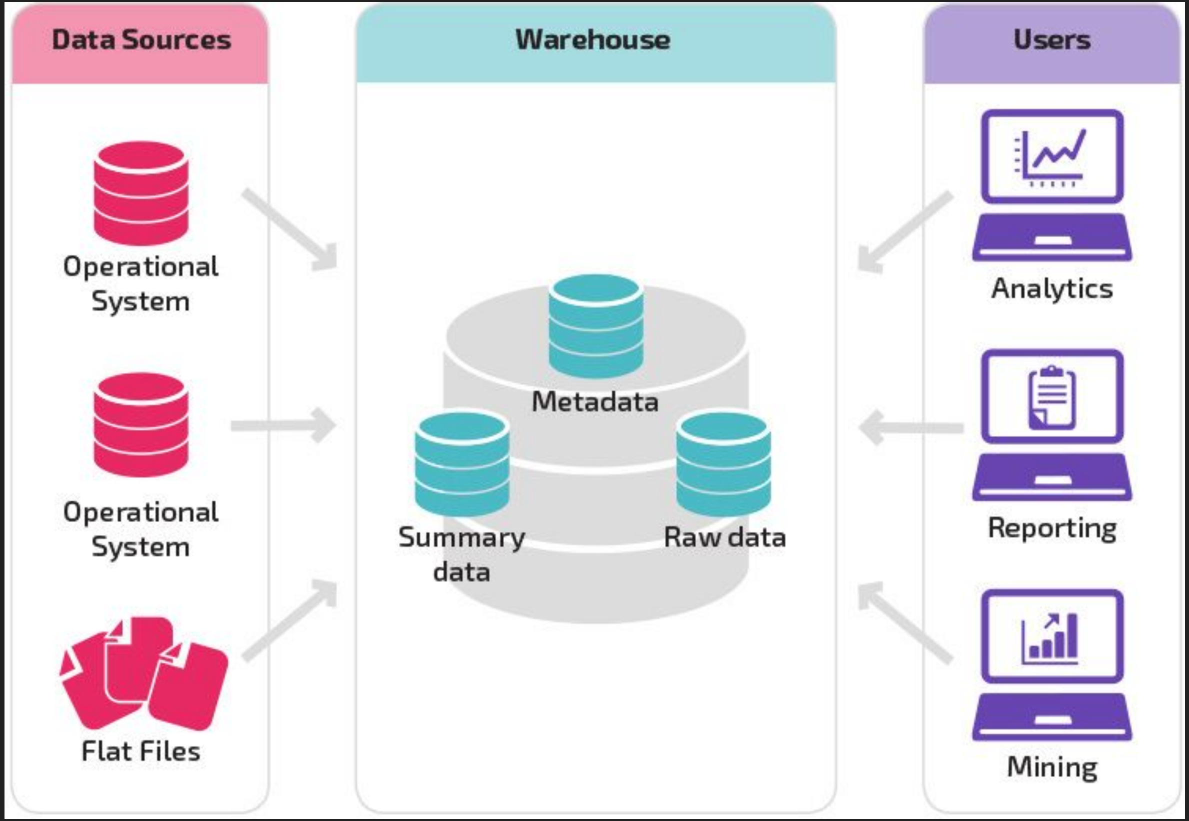
SQL

Data Warehouses and ETL

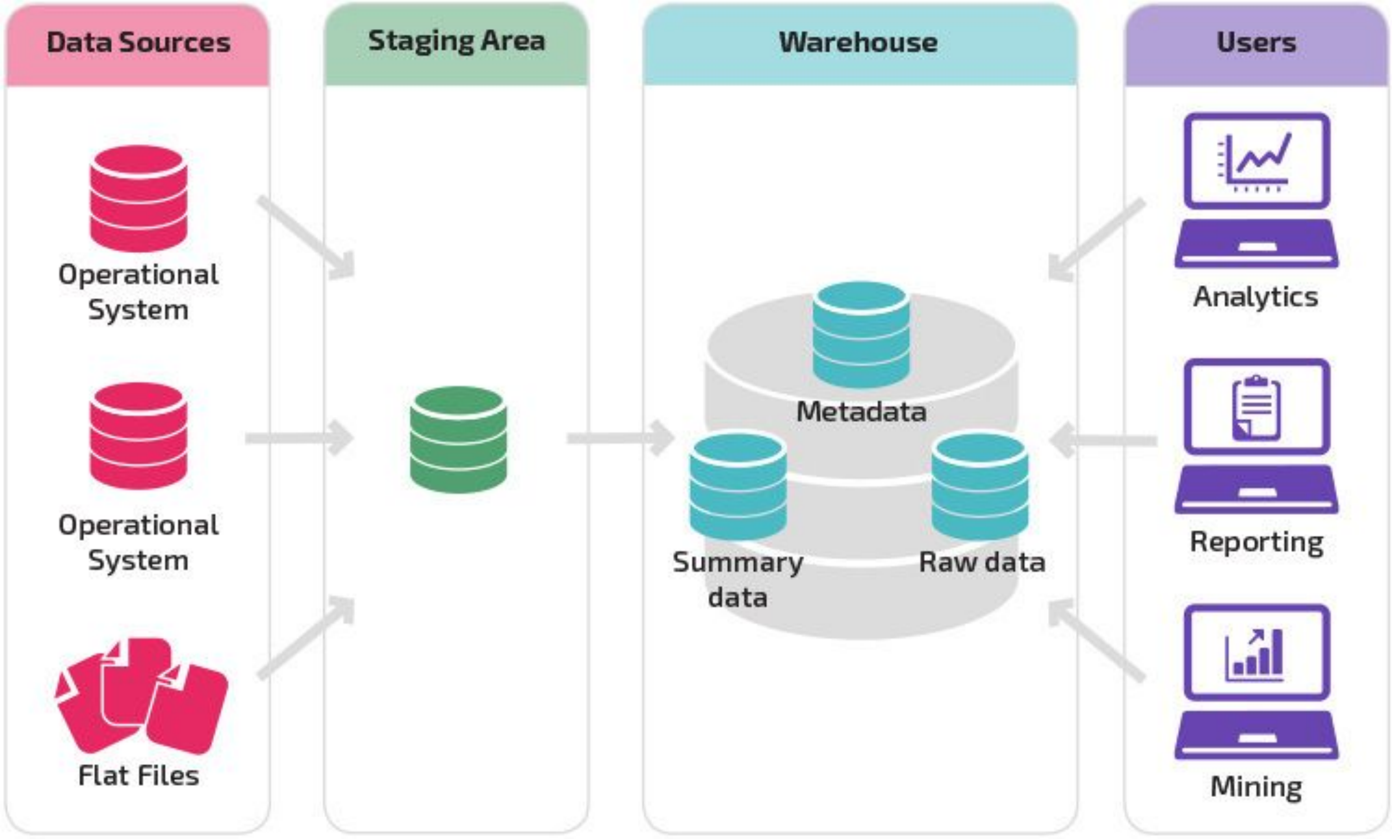
What is a Data Warehouse

A data warehouse is a system that pulls together data from many different sources within an organization for reporting and analysis. The reports created from complex queries within a data warehouse are used to make business decisions.

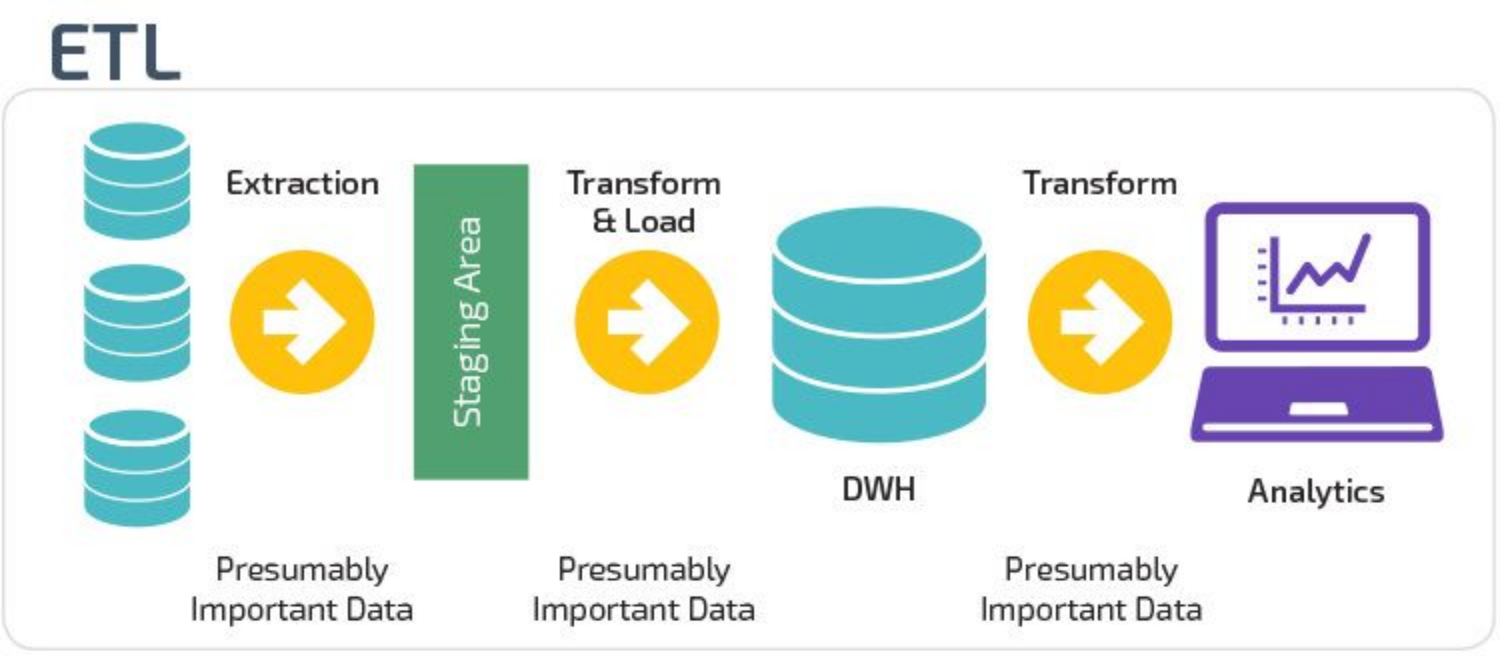
Data Warehouse Architecture Example



Data Warehouse Architecture with Staging Area Example



ETL - Extract, Transformation, Load


ETL
Extraction
Staging Area
Transform & Load
DWH
Transform
Analytics
Presumably Important Data
Presumably Important Data
Presumably Important Data

Cloud Data Warehouse

1. Amazon Redshift
2. Google BigQuery
3. Snowflake
4. Microsoft Azure SQL Data Warehouse
5. And more ….

amazon
REDSHIFT

Microsoft Azure

snowflake

Google
BigQuery

Data Modeling

1NF - 1st Normal Form
2NF - 2nd Normal Form
3NF - 3rd Normal Form

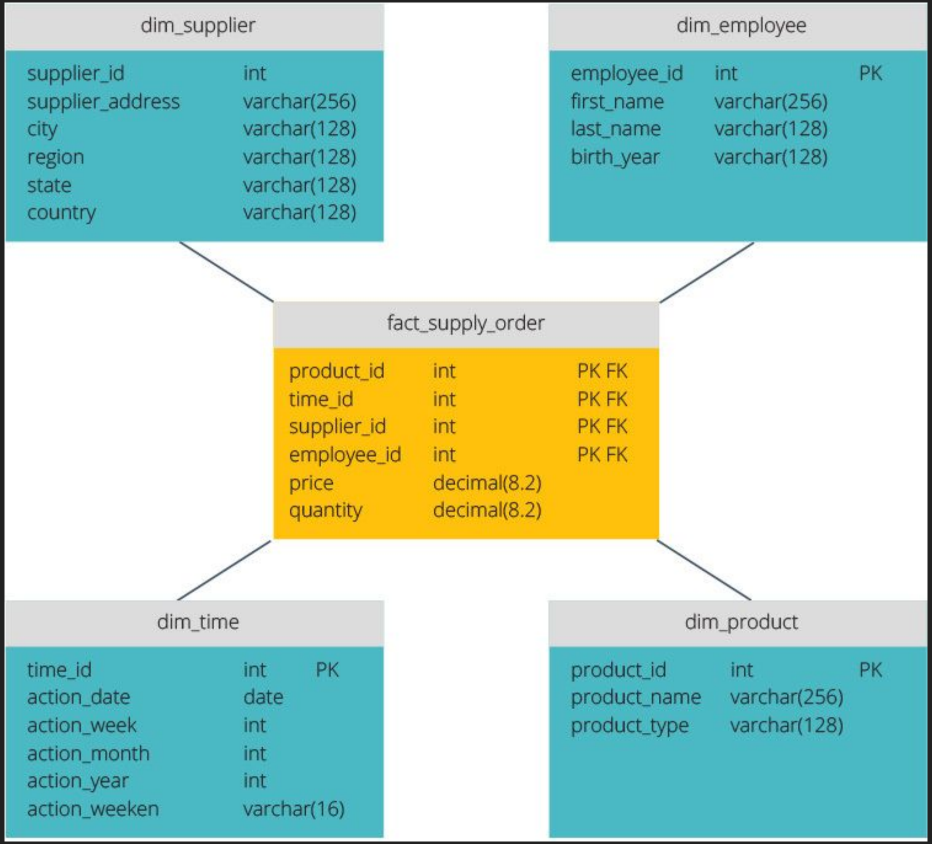
Data Modeling

1NF - 1st Normal Form
2NF - 2nd Normal Form
3NF - 3rd Normal Form

A dimensional model is a data structure technique optimized for Data warehousing tools. The concept of Dimensional Modelling was developed by Ralph Kimball and is comprised of "fact" and "dimension" tables.

A Dimensional model is designed to read, summarize, analyze numeric information like values, balances, counts, weights, etc. in a data warehouse. In contrast, relation models are optimized for addition, updating and deletion of data in a real-time Online Transaction System (aka OLTP)

Dimensional Modeling Example: Star Schema



dim_supplier

supplier_id	int
supplier_address	varchar(256)
city	varchar(128)
region	varchar(128)
state	varchar(128)
country	varchar(128)

dim_employee

employee_id	int	PK
first_name	varchar(256)	
last_name	varchar(128)	
birth_year	varchar(128)	

fact_supply_order

product_id	int	PK FK
time_id	int	PK FK
supplier_id	int	PK FK
employee_id	int	PK FK
price	decimal(8.2)	
quantity	decimal(8.2)	

dim_time

time_id	int	PK
action_date	date	
action_week	int	
action_month	int	
action_year	int	
action_weeken	varchar(16)	

dim_product

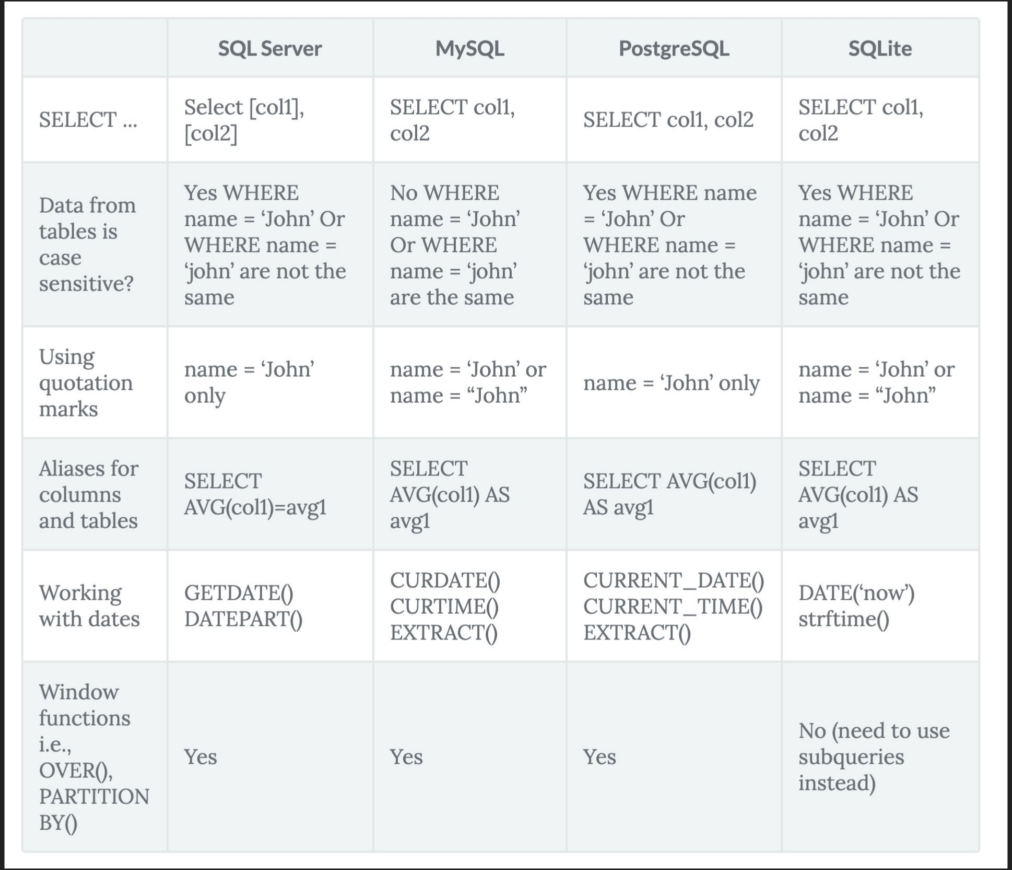
product_id	int	PK
product_name	varchar(256)	
product_type	varchar(128)	

SQLAlchemy

SQL + Python

Database - Differences in SQL implementation

http://troels.arvin.dk/db/rdbms/



	SQL Server	MySQL	PostgreSQL	SQLite
SELECT …	Select [col1], [col2]	SELECT col1, col2	SELECT col1, col2	SELECT col1, col2
Data from tables is case sensitive?	Yes WHERE name = ‘John’ Or WHERE name = ‘john’ are not the same	No WHERE name = ‘John’ Or WHERE name = ‘john’ are the same	Yes WHERE name = ‘John’ Or WHERE name = ‘john’ are not the same	Yes WHERE name = ‘John’ Or WHERE name = ‘john’ are not the same
Using quotation marks	name = ‘John’ only	name = ‘John’ or name = “John”	name = ‘John’ only	name = ‘John’ or name = “John”
Aliases for columns and tables	SELECT AVG(col1)=avg1	SELECT AVG(col1) AS avg1	SELECT AVG(col1) AS avg1	SELECT AVG(col1) AS avg1

Working with dates	GETDATE()
	DATEPART()
		CURDATE()
		CURTIME()
		EXTRACT()
			CURRENT_DATE()
			CURRENT_TIME()
			EXTRACT()
				DATE(‘now’)
				strftime()

Window functions i.e., OVER(), PARTITION BY()	Yes	Yes	Yes	No (need to use subqueries instead)

SQLite

SQLite is a self-contained, file-based, and fully open-source RDBMS known for its portability, reliability, and strong performance even in low-memory environments. Its transactions are ACID-compliant, even in cases where the system crashes or undergoes a power outage.

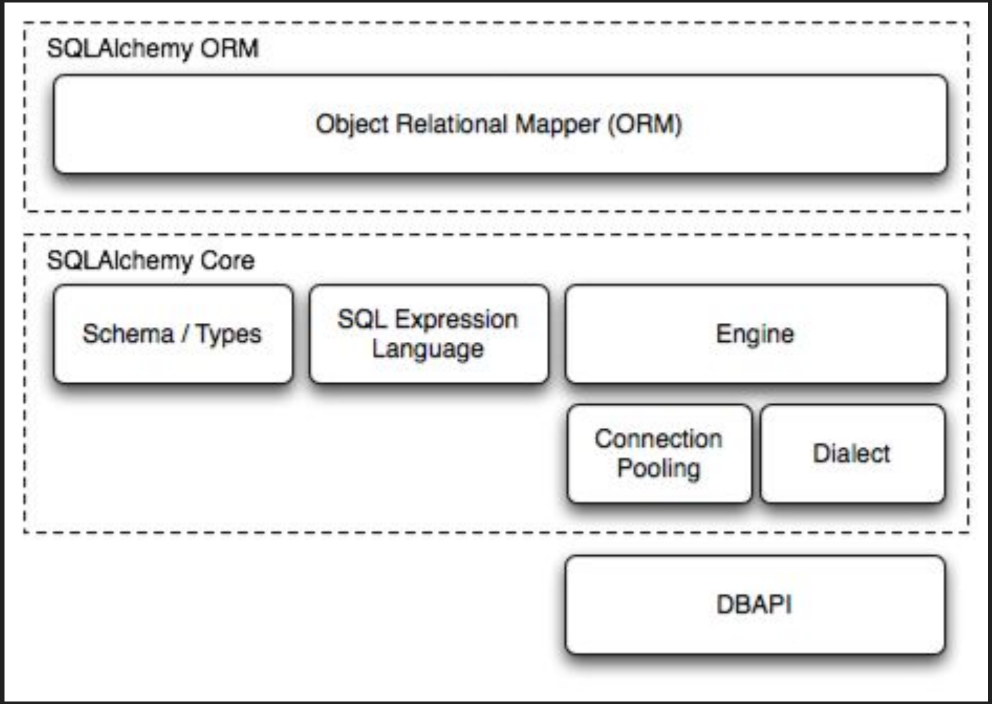
The SQLite project's website describes it as a "serverless" database. Most relational database engines are implemented as a server process in which programs communicate with the host server through an interprocess communication that relays requests. With SQLite, though, any process that accesses the database reads from and writes to the database disk file directly.

SQLAlchemy - ORM

SQLAlchemy is a Python ORM - Object Relational Mapper - for SQL databases

https://www.sqlalchemy.org

https://docs.sqlalchemy.org/en/13/dialects/



SQLAlchemy

- ORM (Object Relational Mapping): Creates an abstraction layer that represents records that are stored in database tables as instances of class.
- Two Primary components:
  - Core - responsible for interaction with the database and performing SQL commands
  - ORM - Works on top of Core and implements Object-Relational Mapping

Core Components of SQLAlchemy

- Dialects
    - Used to communicate with a specific database driver i.e. Oracle, MS SQL, PostgreSQL and MySQL
- Connection-pooling
    - Responsible for establishing connections to databases using a dialect, managing connection pools, and providing a connection API to the engine
- Engine
    - Represents the database for other components that perform SQL commands (starting point)
- SQL expression language
    - An abstraction layer that translates high-level API calls to SQL that engine could execute
- Schema and Types
    - Objects that define the logical data model

SQLAlchemy Syntax

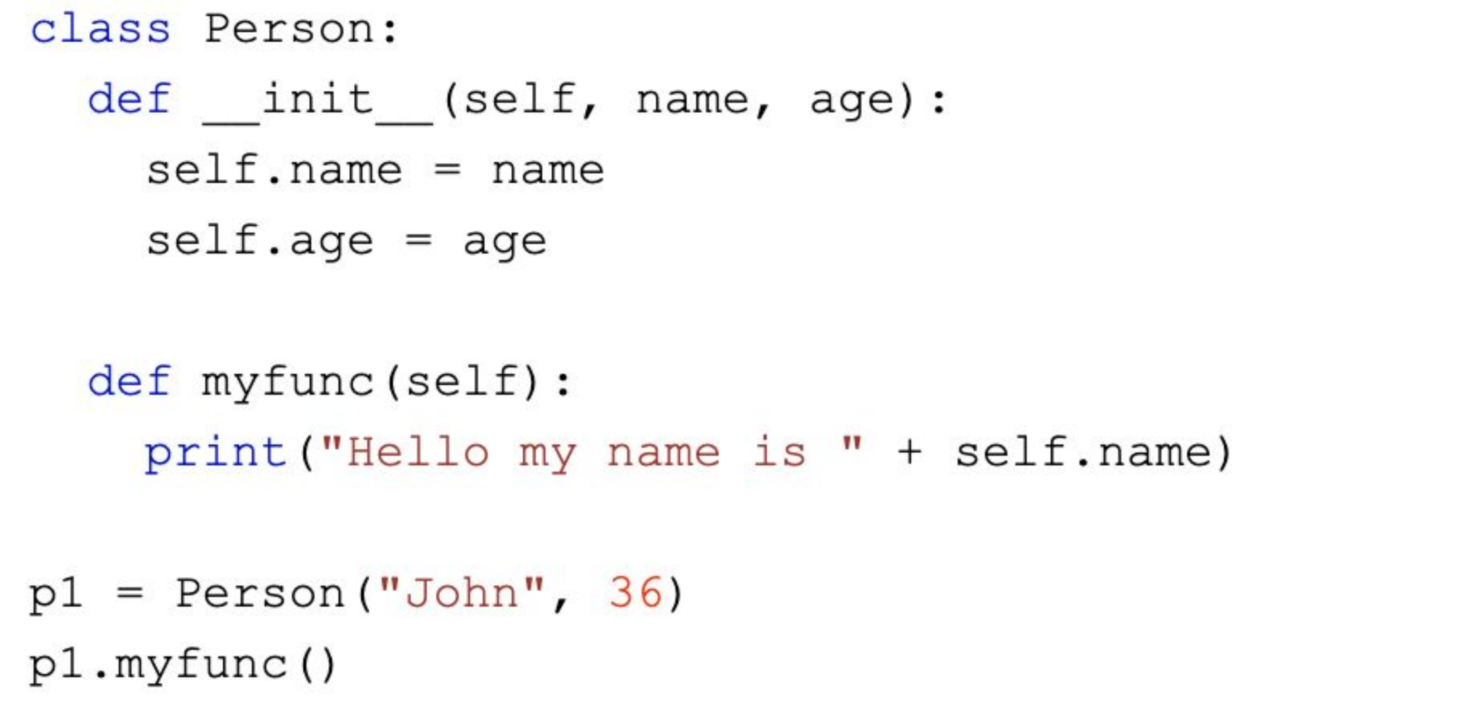
```
engine = create_engine()
conn = engine.connect()
Base = declarative_base()

class Dog(Base):
    __tablename__ = 'dogs'
    id   = Column(Integer, primary_key=True)
    name = Column(String(255))
    color = Column(String(255))
    age   = Column(Integer)
```

Python

Classes & Methods

Python - Classes

https://docs.sqlalchemy.org/en/13/orm/tutorial.html



```python
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```