

UNIVERZITA J. E. PURKYNĚ V ÚSTÍ NAD LABEM

Přírodovědecká fakulta

Přírodovědecká fakulta,
Univerzita J. E. Purkyně v Ústí nad Labem

Nasa Exoplanets Data Lakehouse

Autor: Patrik Poklop

9. května 2025

Obsah

1	Úvod do seminární práce	2
2	Výběr databázového systému	2
3	Výběr databáze	2
4	Vytváření tabulky exoplanet v db systému	2
4.1	ERD hlavní tabulky	4
4.2	Zobrazení tabulek a informací	5
5	Vytvoření dimenzionálních tabulek	7
6	ERD diagram	13
7	Uložení DLH databáze do souboru	14
8	Práce s Lakehouse databází	15
9	Query vyhledávání	16
10	Ukládání a načítání selectů	18
10.1	Načtení selectů	18
11	Statistické grafy	19
11.1	Heatmapa zobrazující počty exoplanet podle roku objevu a metody detekce	19
11.2	Graf s časovou řadou počtu objevených exoplanet podle roku objevu	20
11.3	Sloupcový graf počtu objevených exoplanet podle roku objevu s kategoriemi typů exoplanet	22
11.4	Sloupcový graf počtu objevených exoplanet podle roku objevu s kategoriemi typů metod detekce	23
11.5	Sloupcový graf počtu objevených exoplanet podle množství jejich typu s kategoriemi jejich vzdálenosti	25
11.6	Sloupcový graf objevených exoplanet podle typu a délky orbitální periody .	26

1 Úvod do seminární práce

Vybral jsem si seminární práci na téma exoplanet a informací o nich, kde vytvořím k nim DLH. Práce je udělaná v programovacím jazyce Python. Práce je rozdělena na dvě části programovacího kódu v pythonu. První "DLH_exoplanets_commit.py" je pro vytvoření databáze a druhý "DLH_exoplanets_storages_use.py" je pro načítání dat z Lakehouse souborů, práce s nimi a vytváření statistických grafů k nim. Repozitář se všemi těmito kódy a soubory je k dispozici na:

https://github.com/imang212/NASA_exoplanets_OLAP)

2 Výběr databázového systému

Pro svou seminární práci jsem si vybral DuckDB databázový systém (<https://duckdb.org/>), který umožňuje vytváření kombinovaného DLH databázového systému, kde je možnost vytváření dimenzí dat a datových jezer s daty. DuckDB je zároveň jednoduchý, rychlý a perfektní pro vytváření menších nebo středních DLH. Hlavní důvod proč jsem si vybral tenhle systém, že je u něho možnost práce v Pythonu i v dalších programovacích jazycích.

3 Výběr databáze

Pro svou seminární práci jsem si vybral vypracovanou tabulku NASA Exoplanet od autora Aditya Mishra ML, která obsahuje data z amerického vládního NASA archivu s vylepšenějším popisem některých vlastností na stránkách kaggle.com v csv formátu

<https://www.kaggle.com/datasets/adityamishraml/nasaexoplanets>

Ze stránek oficiálního amerického vládního archivu exoplanet NASA jsem si stáhl druhou tabulku objevených exoplanet, která je také k dispozici v csv formátu na jejich stránkách.

<https://exoplanetarchive.ipac.caltech.edu>.

4 Vytváření tabulky exoplanet v db systému

Nejdřív si vytvořím první python soubor "DLH_exoplanets_storages_use.py", který bude sloužit pro prvotní vytvoření DWH databáze s hvězdicových schématem a jejího uložení do DLH souborů.

Zobrazíme si csv soubory, abychom měli přehled o tom, co tabulky obsahují za data.

```
db = duckdb.read_csv("adityamishraml/nasaexoplanets/versions/2/cleaned_5250.csv")
duckdb.sql("SELECT * FROM 'db'").show()

db2 = duckdb.read_csv("PS_2025.04.28_06.13.44.csv")
duckdb.sql("SELECT * FROM 'db'").show()
```

Obrázek 1: Zobrazení tabulek pomocí duckdb a pandas knihovny

Příkazem show používám duckdb i pandas knihovnu pro zobrazení dataframu dat v tabulce, poté máme přibližný přehled o struktuře tabulky.

Pro připojení k DuckDB budu potřebovat nainstalované knihovny pro práci v pythonu "duckdb" a pro zobrazování grafů "pandas".

```
2 # pip install duckdb
3 import duckdb
4 import pandas as pd
5
6 # connect to DuckDB
7 con = duckdb.connect()
```

Obrázek 2: Ukázka kódu připojení v Pythonu

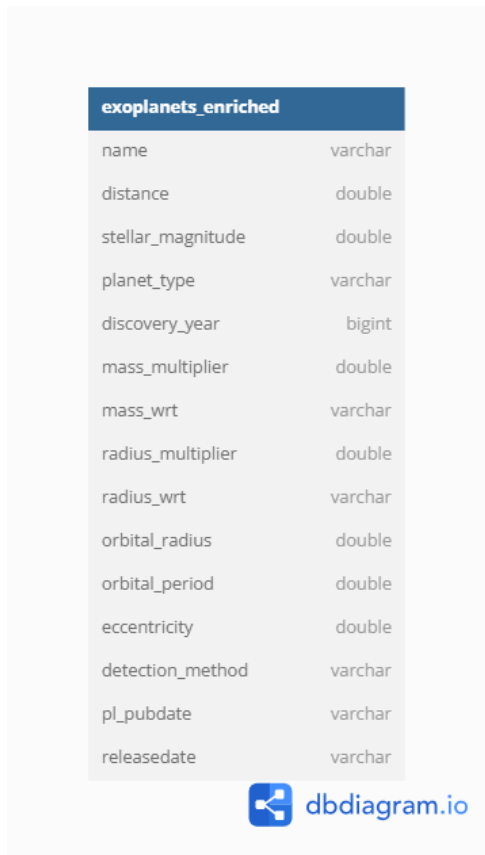
Vytvoříme tabulku, kde si vezmu všechno z první tabulky a potom jí spojím s potřebnými sloupci pro práci s časovými dimenzemi z druhé tabulky.

```
15 con.execute("""
16     CREATE TABLE exoplanets AS
17     SELECT e.*, n.pl_pubdate, n.releasedate
18     FROM read_csv_auto('adityamishram1/nasaexoplanets/versions/2/cleaned_5250.csv') e
19     LEFT JOIN read_csv_auto("PS_2025.04.28_06.13.44.csv") n ON LOWER(e.name) = LOWER(n.pl_name)
20 """)
21
```


Obrázek 3: Vytvoření hlavní tabulky

Tímto příkazem jsem vytvořil hlavní tabulku exoplanet v DuckDB spojením všech sloupců z první tabulky a poté tabulku spojím s výběrem sloupců obsahujících časové údaje pro práci s časovými dimenzemi z druhé tabulky. Druhou tabulku připojím LEFT JOINEM, který ignoruje případné NULL hodnoty na levé straně a vrátí všechny hodnoty z levé strany. Následně jsem z toho vytvořil tabulku "exoplanets".

4.1 ERD hlavní tabulky



exoplanets_enriched	
name	varchar
distance	double
stellar_magnitude	double
planet_type	varchar
discovery_year	bigint
mass_multiplier	double
mass_wrt	varchar
radius_multiplier	double
radius_wrt	varchar
orbital_radius	double
orbital_period	double
eccentricity	double
detection_method	varchar
pl_pubdate	varchar
releasedate	varchar

 dbdiagram.io

Obrázek 4: ERD diagram

Tento ERD diagram vytvořený pomocí webové api (dbdiagram.io) nám zobrazuje hlavní sloupce po spojení a vytvoření tabulky a datové typy k nim.

4.2 Zobrazení tabulek a informací

Zobrazíme si část tabulky pomocí pandas knihovny pro zobrazení dataframů, kde příkaz vezme začátek a konec z obou stran tabulky.

```
22 con.table("exoplanets").show()
```

Obrázek 5: Příkaz k zobrazení informací o tabulce

name varchar	distance double	stellar_magnitude double	planet_type varchar	discovery_year int64	mass_multiplier double	...	radius_earth varchar	orbital_radius double	orbital_period double	eccentricity double	detection_method varchar	pl_publicdate varchar	releasedate varchar
1RXS J160929.1-210...	454.0	12.618	Gas Giant	2008	8.0	--	Jupiter	330.0	6505.9	0.0	Direct Imaging	2015-03	2015-04-01
2M0437 b	419.0	16.186	Gas Giant	2021	4.0	--	Jupiter	118.0	3118.6	0.0	Direct Imaging	2021-10	2021-12-13
2MASS J01033563-55...	154.0	15.788	Gas Giant	2013	13.0	--	Jupiter	84.0	1767.2	0.0	Direct Imaging	2013-05	2020-09-03
2MASS J01225993-24...	110.0	14.244	Gas Giant	2013	24.5	--	Jupiter	52.0	593.2	0.0	Direct Imaging	2013-09	2014-05-14
2MASS J02192210-39...	131.0	NULL	Gas Giant	2015	13.9	--	Jupiter	156.0	5878.1	0.0	Direct Imaging	2015-06	2015-05-14
2MASS J04414489+23...	393.0	NULL	Gas Giant	2010	7.5	--	Jupiter	15.0	411.0	0.0	Direct Imaging	2010-05	2014-05-14
2MASS J12073346-39...	210.0	20.15	Gas Giant	2004	5.0	--	Jupiter	55.0	2885.9	0.0	Direct Imaging	2004-10	2016-02-25
2MASS J19383260+46...	1293.0	12.651	Gas Giant	2015	1.9	--	Jupiter	0.92	1.1115674	0.33	Eclipse Timing Var...	2022-04	2022-04-19
2MASS J22362452+47...	227.0	12.368	Gas Giant	2016	12.5	--	Jupiter	230.0	4585.7	0.0	Direct Imaging	2017-01	2016-11-10
BD+03 2562 b	3492.0	9.577	Gas Giant	2017	6.4	--	Jupiter	1.3	1.3193703	0.2	Radial Velocity	2017-10	2017-06-15
.	--
.	--
.	--
V0391 Pegasi b	3901.0	14.769	Gas Giant	2007	3.2	--	Jupiter	1.7	3.2	0.0	Pulsation Timing V...	NULL	NULL
V1298 Tauri b	353.0	10.115	Gas Giant	2019	0.236	--	Jupiter	0.1688	0.05598221	0.29	Transit	NULL	NULL
V1298 Tauri c	353.0	10.115	Neptune-like	2019	26.7	--	Jupiter	0.0025	0.022458376	0.43	Transit	NULL	NULL
V1298 Tauri d	353.0	10.115	Neptune-like	2019	33.6	--	Jupiter	0.1083	0.03394935	0.21	Transit	NULL	NULL
V1298 Tauri e	353.0	10.115	Gas Giant	2019	0.179	--	Jupiter	0.388	0.16427104	0.57	Transit	NULL	NULL
V830 Tauri b	424.0	12.23	Gas Giant	2016	0.7	--	Jupiter	0.057	0.013415469	0.0	Radial Velocity	NULL	NULL
Xi Aquilae b	183.0	4.70964	Gas Giant	2007	2.8	--	Jupiter	0.68	0.374538	0.0	Radial Velocity	NULL	NULL
YZ Ceti b	12.0	12.074	Terrestrial	2017	0.7	--	Earth	0.01634	0.0054757018	0.06	Radial Velocity	NULL	NULL
YZ Ceti c	12.0	12.074	Super Earth	2017	1.14	--	Earth	0.02156	0.008487337	0.0	Radial Velocity	NULL	NULL
YZ Ceti d	12.0	12.074	Super Earth	2017	1.89	--	Earth	0.02851	0.012867898	0.07	Radial Velocity	NULL	NULL
5250 rows (20 shown)												15 columns (13 shown)	

Obrázek 6: Ukázka informací zobrazených v mé tabulce

Z této tabulky můžeme mít přehled o dané tabulce v db, kolik v ní je sloupců, kolik v ní je položek a jak přibližně vypadají názvy některých údajů uložených v tabulce. Dále si informace o proměnných v tabulce pomocí příkazu fetchdf().

```
23 print(con.execute("DESCRIBE exoplanets").fetchdf())
```

Obrázek 7: Příkaz k zobrazení informací o proměnných v dané tabulce

	column_name	column_type	null	key	default	extra
0	name	VARCHAR	YES	None	None	None
1	distance	DOUBLE	YES	None	None	None
2	stellar_magnitude	DOUBLE	YES	None	None	None
3	planet_type	VARCHAR	YES	None	None	None
4	discovery_year	BIGINT	YES	None	None	None
5	mass_multiplier	DOUBLE	YES	None	None	None
6	mass_wrt	VARCHAR	YES	None	None	None
7	radius_multiplier	DOUBLE	YES	None	None	None
8	radius_wrt	VARCHAR	YES	None	None	None
9	orbital_radius	DOUBLE	YES	None	None	None
10	orbital_period	DOUBLE	YES	None	None	None
11	eccentricity	DOUBLE	YES	None	None	None
12	detection_method	VARCHAR	YES	None	None	None
13	pl_pubdate	VARCHAR	YES	None	None	None
14	releasedate	VARCHAR	YES	None	None	None

Obrázek 8: Ukázka zobrazených informací o proměnných v mé tabulce

Nyní máme přehled o všech proměnných, co obsahuje tabulka a o jejich vlastnostech. o jaký je datový typ se jedná, jestli je nulová, jestli je daná proměnná klíč nebo jestli je výchozí či extra.

5 Vytvoření dimenzionálních tabulek

Nyní si vytvoříme 9 dimenzionálních tabulek.:

- `dim_planet_type` - obsahuje rozdělení na typy planet
- `dim_detection_method` - Obsahuje metody detekce planety rozdělení podle dimenzionálních kategorií.
- `dim_stellar_type` - Obsahuje kategorie zářivosti hvězdy dané planety podle její vzdálenosti od hvězdy Very Bright, Bright, Moderate, Dim a Very Dim.
- `dim_mass_category` - Obsahuje hmotnosti planet rozdělené do kategorií Very Low Mass, Low Mass, Medium Mass a High Mass.
- `dim_distance_category` -obsahuje vzdálenosti planet rozdělené do kategorií Very Close(<10 ly), Close (<100 ly), Medium (<1000 ly) a Far (>1000 ly).
- `dim_orbit_category` - Obsahuje planety rozdělené podle vzdálenosti jejich oběhu Very Short, Short, Moderate a Long.
- `dim_brightness_category` - Obsahuje planety rozdělené podle jasu Very Bright, Bright, Dim a Very Dim.
- `dim_discovery_era` - Obsahuje planety rozdělené podle éry objevení: <2000, Early 21st Century, Kepler Era a Modern Era.
- `dim_date` - Obsahuje časová data o zveřejnění planet, kde celé datum je rozdělené na rok, měsíc, den, název měsíce a název dne.

Ukázka kódu vytvoření dimenzionálních tabulek v DuckDB přes Python.:

```
## vytvoreni dim tabulek
# dim_planet_type
con.execute("""
CREATE_OR_REPLACE_TABLE dim_planet_type AS
SELECT_ROW_NUMBER() OVER() AS_planet_type_id, planet_type
FROM (
SELECT_DISTINCT planet_type
FROM exoplanets
WHERE planet_type IS NOT NULL
) t;
""")
# dim_detection_method
con.execute("""
CREATE_OR_REPLACE_TABLE dim_detection_method AS
SELECT_ROW_NUMBER() OVER() AS_detection_method_id, detection_method
FROM (
SELECT_DISTINCT detection_method
FROM exoplanets
```



```

WHERE detection_method IS NOT NULL
) ) t;
"""
# dim_stellar_type
con.execute("""
CREATE_OR_REPLACE_TABLE dim_stellar_type AS
SELECT_ROW_NUMBER() OVER() AS stellar_type_id, distance, stellar_magnitu
CASE
WHEN stellar_magnitude < 0 THEN 'very_bright'
WHEN stellar_magnitude BETWEEN 0 AND 2 THEN 'bright'
WHEN stellar_magnitude BETWEEN 2 AND 5 THEN 'moderate'
WHEN stellar_magnitude BETWEEN 5 AND 10 THEN 'dim'
ELSE 'very_dim'
END AS brightness_category
FROM (
SELECT_DISTINCT distance, stellar_magnitude
FROM exoplanets
WHERE distance IS NOT NULL AND stellar_magnitude IS NOT NULL
) ) t;
"""
# dim_mass_category
con.execute("""
CREATE_OR_REPLACE_TABLE dim_mass_category AS
SELECT_ROW_NUMBER() OVER() AS mass_category_id, mass_multiplier,
CASE
WHEN mass_multiplier < 0.1 THEN 'Very_Low_Mass'
WHEN mass_multiplier < 1 THEN 'Low_Mass'
WHEN mass_multiplier < 5 THEN 'Medium_Mass'
WHEN mass_multiplier < 20 THEN 'High_Mass'
ELSE 'Very_High_Mass'
END AS mass_category
FROM (
SELECT_DISTINCT mass_multiplier,
FROM exoplanets
WHERE mass_multiplier IS NOT NULL
) ) t;
"""
# dim_distance_category
con.execute("""
CREATE_OR_REPLACE_TABLE dim_distance_category AS
SELECT_ROW_NUMBER() OVER() AS distance_category_id, distance,
CASE
WHEN distance < 10 THEN 'Very_Close(<10_ly)'
WHEN distance < 100 THEN 'Close(<100_ly)'
WHEN distance < 1000 THEN 'Medium(<1000_ly)'
ELSE 'Far(>1000_ly)'

```

```

        END_AS distance_category
    FROM (
        SELECT DISTINCT distance ,
        FROM exoplanets
        WHERE distance IS NOT NULL
    ) t;
    """
# dim_orbit_category
con.execute("""
    CREATE_OR_REPLACE_TABLE dim_orbit_category AS
    SELECT ROW_NUMBER() OVER () AS orbit_category_id , orbital_period ,
    CASE
        WHEN orbital_period < 10 THEN 'Very Short '
        WHEN orbital_period < 100 THEN 'Short '
        WHEN orbital_period < 1000 THEN 'Moderate '
        ELSE 'Long '
    END AS period_class
    FROM (
        SELECT DISTINCT orbital_period ,
        FROM exoplanets
        WHERE orbital_period IS NOT NULL
    ) t;
    """)
# dim_brightness_category
con.execute("""
    CREATE_OR_REPLACE_TABLE dim_brightness_category AS
    SELECT ROW_NUMBER() OVER () AS brightness_category_id , stellar_magnitud
    CASE
        WHEN stellar_magnitude < 5 THEN 'Very Bright '
        WHEN stellar_magnitude < 10 THEN 'Bright '
        WHEN stellar_magnitude < 15 THEN 'Dim '
        ELSE 'Very Dim '
    END AS brightness_category
    FROM (
        SELECT DISTINCT stellar_magnitude ,
        FROM exoplanets
        WHERE stellar_magnitude IS NOT NULL
    ) t;
    """)
# dim_discovery_era
con.execute("""
    CREATE_OR_REPLACE_TABLE dim_discovery_era AS
    SELECT ROW_NUMBER() OVER () AS discovery_era_id , discovery_year ,
    CASE
        WHEN discovery_year < 2000 THEN '<2000'
        WHEN discovery_year < 2010 THEN 'Early 21st Century'

```

```

        WHEN discovery_year < 2020 THEN 'Kepler_Era'
        ELSE 'Modern_Era'
        END AS discovery_era
    FROM (
        SELECT DISTINCT discovery_year
        FROM exoplanets
        WHERE discovery_year IS NOT NULL
    ) t;
"""
# dim_date
con.execute("""
    CREATE_OR_REPLACE_TABLE dim_date AS
    SELECT
        ROW_NUMBER() OVER () AS date_id,
        CAST(releasedate AS DATE) AS date,
        date_part('year', CAST(releasedate AS DATE)) AS year,
        date_part('month', CAST(releasedate AS DATE)) AS month,
        strftime(CAST(releasedate AS DATE), '%B') AS month_name,
        date_part('day', CAST(releasedate AS DATE)) AS day,
        strftime(CAST(releasedate AS DATE), '%A') AS weekday_name
    FROM (
        SELECT DISTINCT releasedate
        FROM exoplanets
        WHERE releasedate IS NOT NULL
    ) t;
""")

```

Vytvořili jsme dimenzionální tabulky, které pak budou propojeny podle id hodnoty záznamu v tabulce. U některých tabulek je použit CASE, který nám vytvoří sloupce s rozdělením do pojmenovaných kategorií podle hodnoty, která do tabulky vstupuje. Důležité je u těchto tabulek mít u SELECT příkazu DISTINCT, který nám poté vrátí pouze odlišné hodnoty.

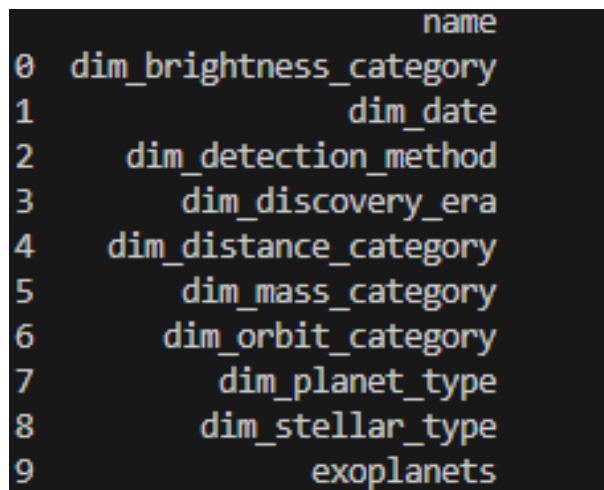
Nyní, když máme vytvořeny dimenzionální tabulky, stačí je namapovat na naší hlavní tabulku exoplanet.

```
# propojeni dimenzionalnich tabulek s tabulkou exoplanet
con.execute("""
    CREATE_OR_REPLACE_TABLE exoplanets AS
    SELECT e.*,
           p.planet_type_id,
           d.detection_method_id,
           s.stellar_type_id,
           m.mass_category_id,
           dc.distance_category_id,
           o.orbit_category_id,
           b.brightness_category_id,
           de.discovery_era_id,
           dt.date_id
    FROM exoplanets_e
    LEFT JOIN dim_planet_type_p ON e.planet_type = p.planet_type
    LEFT JOIN dim_detection_method_d ON e.detection_method = d.detection_me
    LEFT JOIN dim_stellar_type_s ON e.distance = s.distance AND e.stellar_m
    LEFT JOIN dim_mass_category_m ON e.mass_multiplier = m.mass_multiplier
    LEFT JOIN dim_distance_category_dc ON e.distance = dc.distance
    LEFT JOIN dim_orbit_category_o ON e.orbital_period = o.orbital_period
    LEFT JOIN dim_brightness_category_b ON e.stellar_magnitude = b.stellar_m
    LEFT JOIN dim_discovery_era_de ON e.discovery_year = de.discovery_year
    LEFT JOIN dim_date_dt ON CAST(e.releasedate AS DATE) = dt.date;
""")
```

U hlavní tabulky exoplanets jsem si vytvořil ID sloupce ke všem 9 dimenzionálním tabulkám, abych je mohl propojit s hlavní tabulkou exoplanets podle hodnot sloupců dimenzionálních tabulek s hodnotami daného sloupce v tabulce exoplanet, podle kterých potom vytvořím sloupec dousedního klíče pro každý záznam v tabulce exoplanet jeho ID propojí s odpovídajícím ID v dimenzionální tabulce. Zase používám LEFT JOIN, aby to ignorovalo prázdné hodnoty vlevo.

Dále si zkontroluju, jestli se nám všechny tabulky vytvořily.

```
print(con.execute("SHOW TABLES").fetchdf())
```



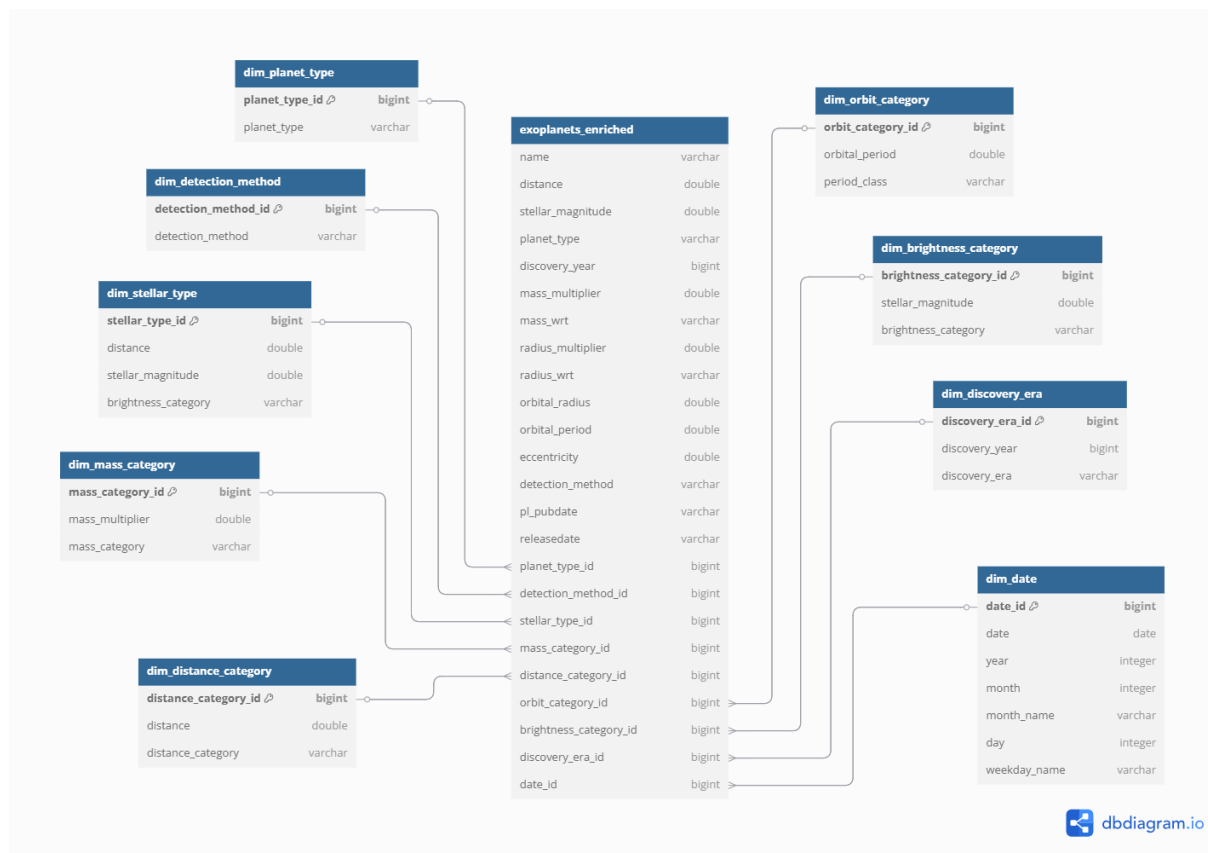
	name
0	dim_brightness_category
1	dim_date
2	dim_detection_method
3	dim_discovery_era
4	dim_distance_category
5	dim_mass_category
6	dim_orbit_category
7	dim_planet_type
8	dim_stellar_type
9	exoplanets

Obrázek 9: Zobrazení všech vytvořených tabulek

Tady můžeme vidět výpis všech vytvořených tabulek v DuckDB systému řazených od 0.

6 ERD diagram

Sestavený výsledný ERD diagram pomocí webové aplikace dbdiagram.io.



Obrázek 10: Výsledný ERD diagram

Nyní můžeme vidět výsledný ERD diagram databáze, který reprezentuje hvězdicové schéma, kde je několik dimenzionálních tabulek připojené k hlavní tabulce pomocí ID. Každá dimenzionální tabulka má primární klíč, který je spojený s daným sousedním klíčem u hlavní tabulky. Teďka se vytvořilo tzv hvězdicové schéma.

7 Uložení DLH databáze do souboru

Aby vytvořené tabulky nebyly ztraceny, uložíme je Parquet(Lake) souborů, kde se nám vytvoří soubory pro Lakehouse úložiště. Všechny je uložíme do nově vytvořené složky "dimensions". Ukládám do složky, aby se dalo lépe vyznat v souborech.

```
import os
os.makedirs('dimensions', exist_ok=True)
# lakehouse storage
con.execute("""
COPY_exoplanets_TO_ 'exoplanets.parquet'_(FORMAT_ 'parquet ');
COPY_dim_planet_type_TO_ 'dimensions/dim_planet_type.parquet'_(FORMAT_ 'parquet ');
COPY_dim_detection_method_TO_ 'dimensions/dim_detection_method.parquet'_(FORMAT_ 'parquet ');
COPY_dim_stellar_type_TO_ 'dimensions/dim_stellar_type.parquet'_(FORMAT_ 'parquet ');
COPY_dim_mass_category_TO_ 'dimensions/dim_mass_category.parquet'_(FORMAT_ 'parquet ');
COPY_dim_distance_category_TO_ 'dimensions/dim_distance_category.parquet'_(FORMAT_ 'parquet ');
COPY_dim_orbit_category_TO_ 'dimensions/dim_orbit_category.parquet'_(FORMAT_ 'parquet ');
COPY_dim_brightness_category_TO_ 'dimensions/dim_brightness_category.parquet'_(FORMAT_ 'parquet ');
COPY_dim_discovery_era_TO_ 'dimensions/dim_discovery_era.parquet'_(FORMAT_ 'parquet ');
COPY_dim_date_TO_ 'dimensions/dim_date.parquet'_(FORMAT_ 'parquet ');
""")
```

V tomto kódu jsem vytvořil, že každé vytvořené tabulce v databázi kopii pomocí příkazu COPY. Dimenzionální tabulky jsem poté uložil ve formátu parquet do nové vytvořené složky "dimensions" pomocí OS knihovny a hlavní tabulku exoplanet jsem uložil také ve formátu parquet do aktuální root složky. Tomuto už by se dalo říkat DLH databáze.

8 Práce s Lakehouse databází

Začneme další část kódu v Pythonu do nového souboru "DLH_exoplanets_storages_use.py", která nám bude pracovat s Lakehouse soubory. Začneme pracovat s uloženými parquet(Lake) soubory. Načtu si je tak, že si ke každému z nich vytvořím VIEW.

```
import duckdb
import pandas as pd
from os import makedirs
```

```
con = duckdb.connect()
```

```
con.execute("""
CREATE_VIEW_dim_planet_type_AS_SELECT_*_FROM_ 'dimensions/dim_planet_type.parquet';
CREATE_VIEW_dim_detection_method_AS_SELECT_*_FROM_ 'dimensions/dim_detection_method.parquet';
CREATE_VIEW_dim_stellar_type_AS_SELECT_*_FROM_ 'dimensions/dim_stellar_type.parquet';
CREATE_VIEW_dim_mass_category_AS_SELECT_*_FROM_ 'dimensions/dim_mass_category.parquet';
CREATE_VIEW_dim_distance_category_AS_SELECT_*_FROM_ 'dimensions/dim_distance_category.parquet';
CREATE_VIEW_dim_orbit_category_AS_SELECT_*_FROM_ 'dimensions/dim_orbit_category.parquet';
CREATE_VIEW_dim_brightness_category_AS_SELECT_*_FROM_ 'dimensions/dim_brightness_category.parquet';
CREATE_VIEW_dim_discovery_era_AS_SELECT_*_FROM_ 'dimensions/dim_discovery_era.parquet';
CREATE_VIEW_dim_date_AS_SELECT_*_FROM_ 'dimensions/dim_date.parquet';
CREATE_VIEW_exoplanets_AS_SELECT_*_FROM_ 'exoplanets.parquet';
""")
```

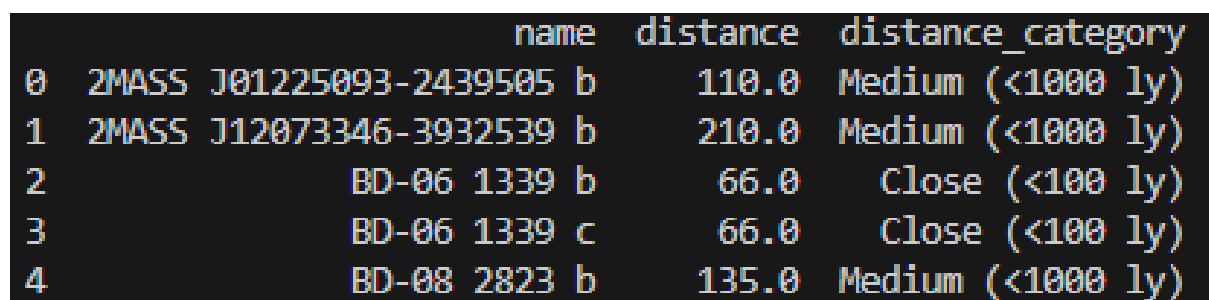
Tímto příkazem se zase vybrali všechny tabulky a vytvořili se z nich views, abychom s nimi mohli pracovat.

9 Query vyhledávání

Nyní si nad načtenými daty vyzkoušíme query vyhledávání.

#zobrazení planet se vzdálenostmi a jejich kategoriemi

```
print(con.execute("""
        SELECT_e.name,_e.distance,_dc.distance_category
        FROM_exoplanets_e
        JOIN_dim_distance_category_dc_ON_e.distance_category_id=_dc.distance_category_id
    """).df().head())
```

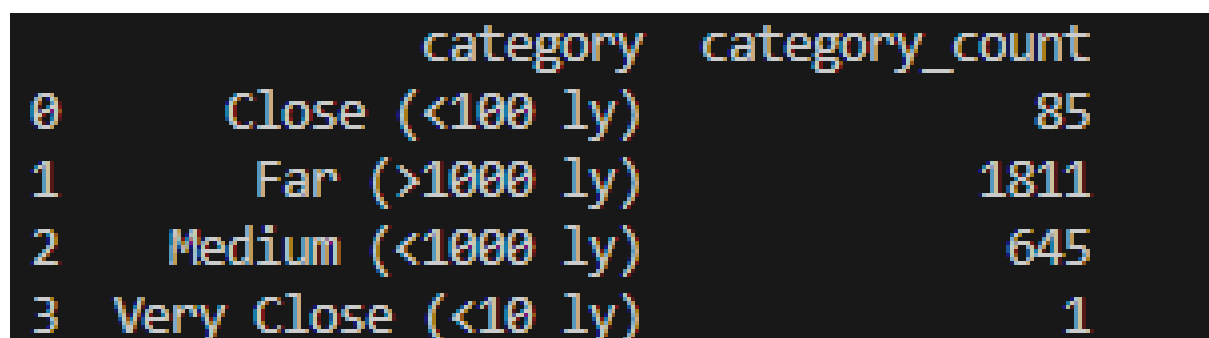


	name	distance	distance_category
0	2MASS J01225093-2439505 b	110.0	Medium (<1000 ly)
1	2MASS J12073346-3932539 b	210.0	Medium (<1000 ly)
2	BD-06 1339 b	66.0	Close (<100 ly)
3	BD-06 1339 c	66.0	Close (<100 ly)
4	BD-08 2823 b	135.0	Medium (<1000 ly)

Obrázek 11: Ukázka tabulky

Select pro zobrazení hlavičky planet s jejich kategoriemi vzdáleností, kde z tabulky spojenou s dimenzionální tabulkou kategorií vzdáleností, vyberu názvy planet, vzdálenost z tabulky exoplanet a kategorii vzdálenosti z dimenzionální tabulky s kategoriemi vzdáleností. Na obrázku je vidět ukázka zobrazení dat z tohoto selectu.

```
print(con.execute("""
        SELECT_dc.distance_category_as_category,_count(dc.distance_category)
        FROM_dim_distance_category_as_dc
        GROUP_BY_dc.distance_category
        ORDER_BY_dc.distance_category
    """).df().head())
```



	category	category_count
0	Close (<100 ly)	85
1	Far (>1000 ly)	1811
2	Medium (<1000 ly)	645
3	Very Close (<10 ly)	1

Obrázek 12: Ukázka tabulky

Pomocí dimenzionální tabulky si spočítám kolik v každé kategorii vzdálenosti je planet. Z dimenzionální tabulky si vyberu kategorii a poté spočítám celkový počet u dané kategorie, aby mi to zobrazilo co chci, nesmím zapomenout to seskupit podle kategorie vzdálenosti a také seřadit podle kategorie vzdálenosti pro lepší přehled. Na obrázku je ukázka zobrazení těchto dat s kategoriemi z tohoto selectu.

10 Ukládání a načítání selectů

Výsledky vytvořených selectů se také dají uložit do parquet(Lake) souborů

```
makedirs( 'results' , exist_ok=True)
con.execute( """
COPY_ (
SELECT
de.discovery_era ,
dm.detection_method ,
COUNT(*) AS num_planets
FROM exoplanets_e
JOIN dim_discovery_era_de ON e.discovery_year = de.discovery_year
JOIN dim_detection_method_dm ON e.detection_method_id = dm.detection_method_id
GROUP BY de.discovery_era , dm.detection_method
ORDER BY num_planets DESC
) TO 'results/planets_era_detection_method.parquet' (FORMAT 'parquet')
""")
```

Tímto jsme uložili kopii selectu spojenom s dimenzionálními tabulkami do lake souboru, kterou jsme uložili do parquet souboru do nové složky s výsledky pojmenované "results". Vytvořila se kopie, kde jsem z hlavní tabulky exoplanet označenou jako e, spojenom s dimenzionálními tabulkami dim_discovery_era rozdělující roky objevů planet do období a dim_detection_method rozdělující metody detekce do dimenzí. Z tabulky dim_discovery_era jsem vybral období objevu a z tabulky dim_detection_method jsem vybral metodu detekce. Všechny planety spočítal pomocí count(*) příkazu. Všechny záznamy jsem seskupil pomocí období objevu a poté metody detekce. Tabulku jsem seřadil podle počtu exoplanet sestupně.

10.1 Načtení selectů

Znovu si načtu z lake souboru jako tabulku zobrazení počtu planet podle roku objevu a metody detekce.

```
df = con.execute("""SELECT * FROM 'results/planets_era_detection_method.parquet'""")
#print(df)
```

U tohoto příkazu stačí, když ze souboru vyberu všechno, převedu to na dataframe a zase s tím mohu pracovat.

11 Statistické grafy

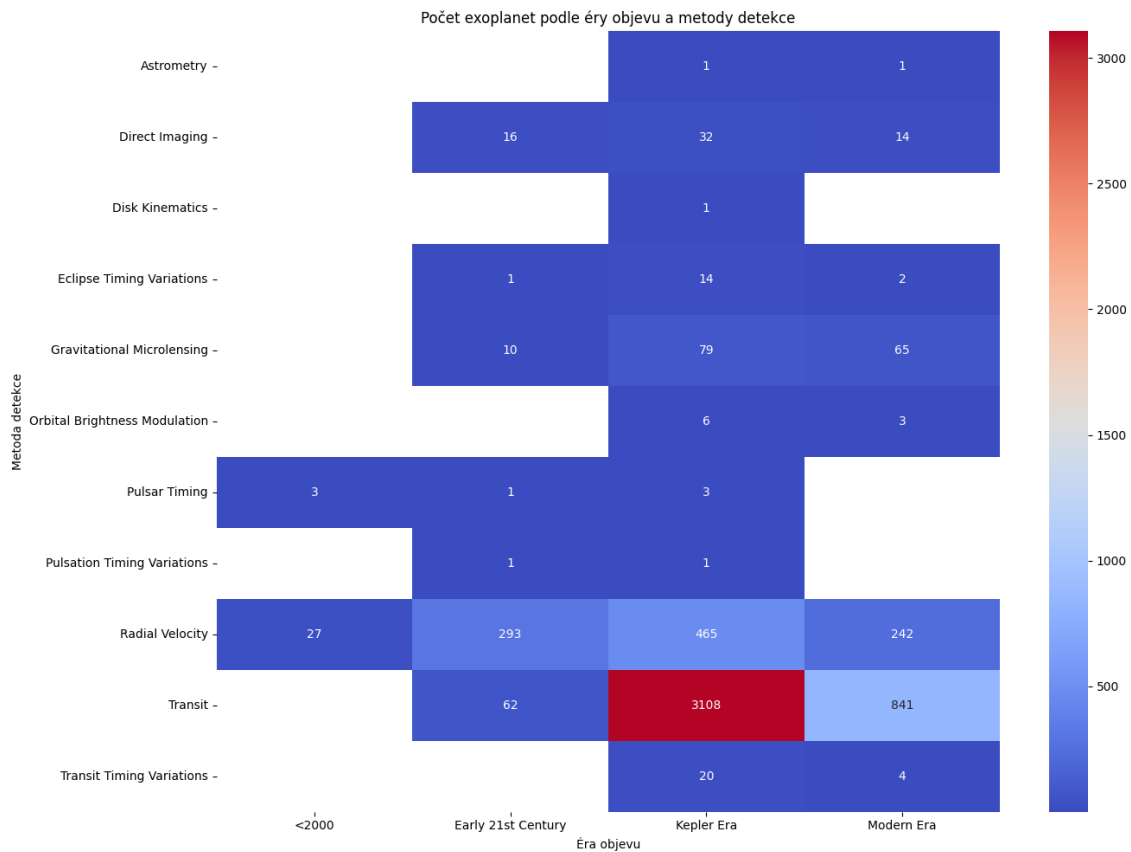
Nyní si ukážeme, jak DLH úložiště je užitečné pro práci s načítání statistických a vytvářením grafů z nich.

11.1 Heatmapa zobrazující počty exoplanet podle roku objevu a metody detekce

```
# Prevod na kontingencni tabulku (pivot)
pivot = df.pivot(index='detection_method', columns='discovery_era', values=
makedirs('graphs', exist_ok=True))

# Heatmapa
plt.figure(figsize=(14, 10))
sns.heatmap(pivot, annot=True, fmt=".0f", cmap="coolwarm")
plt.title("Pocet_exoplanet_podle_ery_objevu_a_metody_detekce")
plt.xlabel("Era_objevu")
plt.ylabel("Metoda_detekce")
plt.tight_layout()
plt.savefig('graphs/exoplanet_era_detection_heatmap.png')
plt.close()
```

Nejdříve si převedu dataframe na kontingenční tabulku neboli pivot pomocí příkazu `pivot()`, kde si označím kterou proměnnou chci jako index, která chci, aby mi reprezentovala sloupce a která mi bude reprezentovat hodnoty v políčkách heatmapy. Následně se mi tabulka pomocí příkazu `pivot()` převede do odpovídajícího formátu pro vykreslení heatmapy. Vytvořím graf heatmapy pomocí `seaborn` knihovny. Výsledné grafy si budu ukládat do složky "graphs".



Obrázek 13: Heatmapa

V heatmapě můžeme vidět, v kterých částech je častější výskyt označených červeně a nebo v kterých naopak ne bez čísla nebo zabarvených tmavě modře.

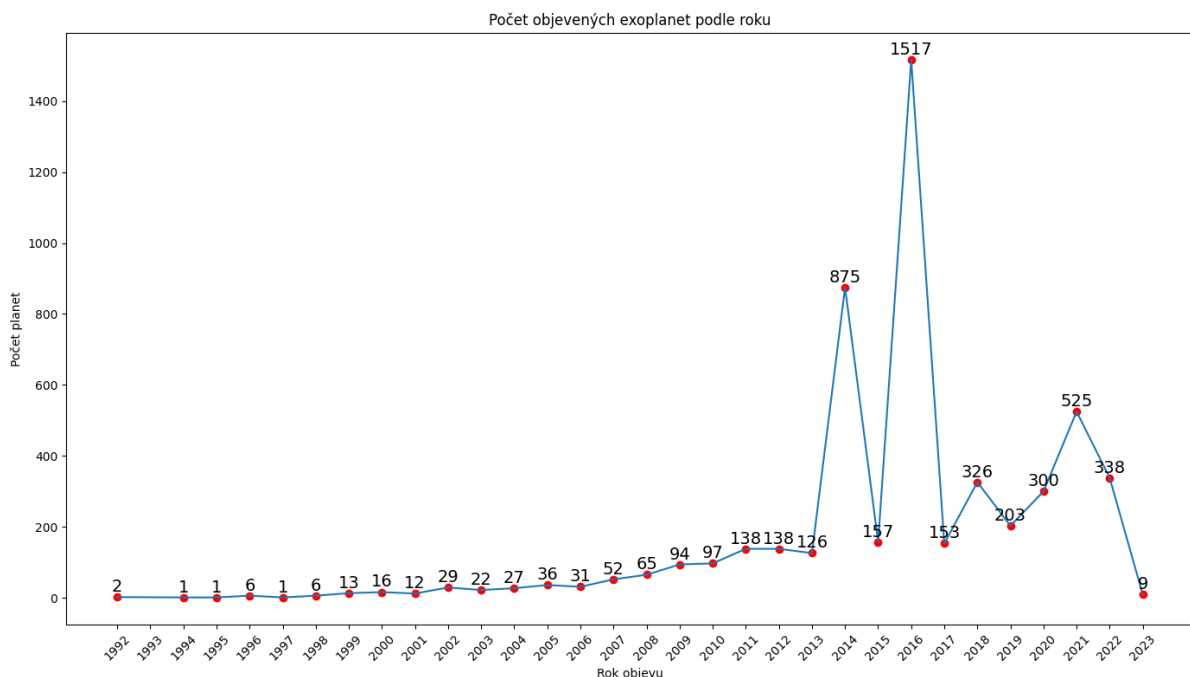
11.2 Graf s časovou řadou počtu objevených exoplanet podle roku objevu

```
con.execute("""
COPY_(
SELECT
de.discovery_year,
COUNT(*) AS num_planets
FROM exoplanets_e
JOIN dim_discovery_era_de ON e.discovery_year = de.discovery_year
JOIN dim_detection_method_dm ON e.detection_method_id = dm.detection_method_id
GROUP BY de.discovery_year
ORDER BY de.discovery_year
) TO 'results/planets_discovery_year_count_timeline.parquet' (FORMAT 'parquet')
""")
df = con.execute("""SELECT * FROM 'results/planets_discovery_year_count_timeline.parquet'""").fetchall()
```

Zase si vytvořím odpovídající select spojený s dimenzionálními tabulkami jako v předešlé kopii selectu pro graf s časovou řadou počtu exoplanet podle roku objevu a potom si zase vyberu všechno z toho souboru a převedu to na dataframe. Nyní přejdeme k vykreslení grafu.

```
plt.figure(figsize=(14, 8))
sns.lineplot(data=df, x='discovery_year', y='num_planets')
plt.scatter(data=df, x='discovery_year', y="num_planets", color='red')
for i, row in df.iterrows():
    plt.text(row['discovery_year'], row['num_planets'] + 5, str(row['num_planets']),
             ha='center', va='bottom', fontsize=14)
plt.title("Pocet objevenych exoplanet podle roku")
plt.xlabel("Rok objevu")
plt.ylabel("Pocet planet")
plt.xticks(np.arange(1992, 2024, 1), rotation=45)
plt.tight_layout()
plt.savefig('graphs/casova_rada_poctu_objevu_exoplanet.png')
plt.close()
```

V tomto kódu jsem zase pomocí seaborn knihovny pomocí příkazu lineplot() jsem vykreslil odpovídající časovou křivku. Nastavil jsem si větší šířku grafu, aby data byla přehlednější. Udělal jsem na grafu body, aby lépe šli vidět vrcholy a ke každému bodu jsem přidal popis s počtem planet. Nezapomněl jsem upřesnit na x ose roky, pro lepší přehled.



Obrázek 14: Časová řada počtů objevených exoplanet v rocích

Tato časová řada nám ukazuje, že nejsilnější trend byl v roce 2016 s hodnotou 1517 a také i u roků kolem 2014-2022 byl vyšší. Naopak nejnižší trend byl u roků 1992-2002,

kde nepřesáhl 20 objevených exoplanet. Zde je vidět, že nám křivka postupně roste od nejnižšího až po vysoké hodnoty kromě posledního roku 2023.

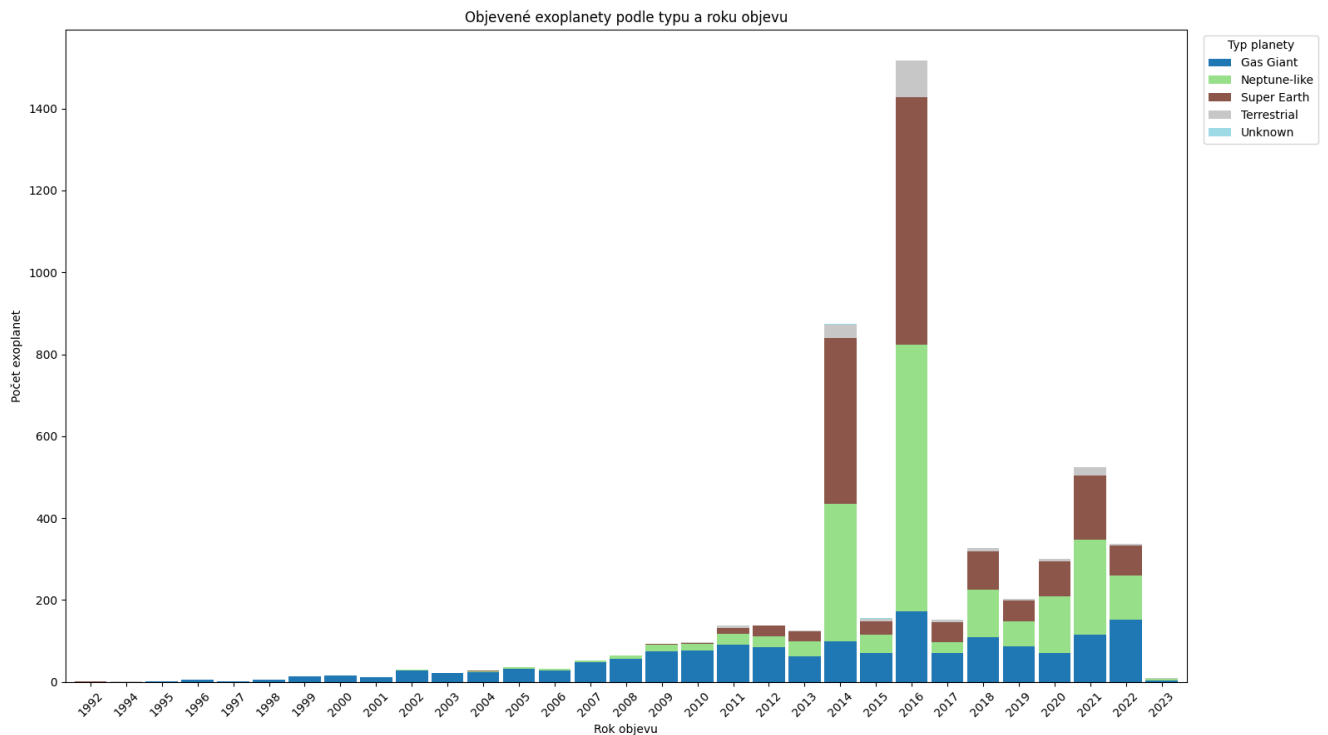
11.3 Sloupcový graf počtu objevených exoplanet podle roku objevu s kategoriemi typů exoplanet

```
con.execute("""
COPY_
SELECT
de.discovery_year ,
e.planet_type ,
COUNT(*)_AS_num_planets
FROM exoplanets_e
JOIN dim_discovery_era_de ON e.discovery_year = de.discovery_year
GROUP BY de.discovery_year , e.planet_type
ORDER BY de.discovery_year
) TO 'results/exoplanet_discovery_count_by_year.parquet' (FORMAT 'parquet')
""")
```

```
df = con.execute("""SELECT_*_FROM 'results/exoplanet_discovery_count_by_year.parquet'
#print(df)
```

```
df_pivot = df.pivot(index='discovery_year', columns='planet_type', values='num_planets')
df_pivot.plot(kind='bar', stacked=True, colormap='tab20', figsize=(16, 9),
plt.title("Objevne_exoplanety_podle_typu_a_roku_objevu")
plt.xlabel("Rok_objevu")
plt.ylabel("Pocet_exoplanet")
plt.xticks(rotation=45)
plt.legend(title='Typ_planety', bbox_to_anchor=(1.01, 1), loc='upper_left')
plt.tight_layout()
plt.savefig("graphs/bar_plot_planet_type_by_year.png")
plt.close()
```

V tomto kódu je postup vytvoření takové sloupcového grafu, kde si zase z dataframu potřebuji udělat pivot, kde ošetřím i nulové hodnoty.



Obrázek 15: Sloupcový graf druhů objevených exoplanet podle roku

Z grafu vyplývá, že první objevené exoplanety do roku 2009 byli skoro všechny plynní obři. Až poté se začaly objevovat v roce 2010+ planety typu Neptunu, superzemě a terestriální planety a nebo dosud neznámé planety. V některých posledních sloupcích 2014+ dokonce u objevených exoplanet jsou více zastoupeny Superzemě a u některých planety typu Neptun než plynní obři. Možná, že to i může souviset s vývojem technologií určených pro objevování exoplanet.

11.4 Sloupcový graf počtu objevených exoplanet podle roku objevu s kategoriemi typů metod detekce

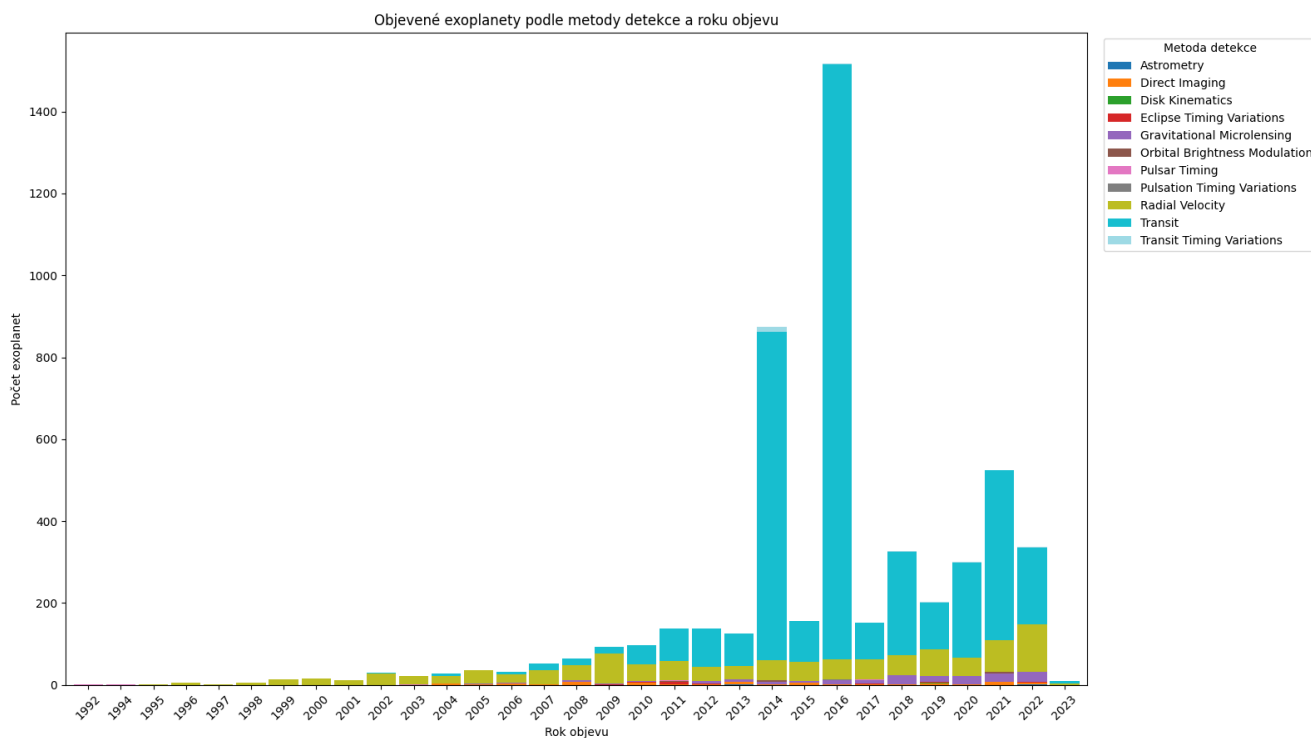
```
con.execute("""
COPY_(
SELECT
de.discovery_year ,
dm.detection_method ,
COUNT(*) AS num_planets
FROM exoplanets_e
JOIN dim_discovery_era_de ON e.discovery_year = de.discovery_year
JOIN dim_detection_method_dm ON e.detection_method_id = dm.detection_method_id
GROUP BY de.discovery_year , dm.detection_method
ORDER BY de.discovery_year
) TO 'results/exoplanet_detection_method_count_by_year.parquet' (FORMAT
""")
```



```
df = con.execute("""SELECT * FROM 'results/exoplanet_detection_method_count'
#print(df)
```

```
df_pivot = df.pivot(index='discovery_year', columns='detection_method', values='count')
df_pivot.plot(kind='bar', stacked=True, colormap='tab20', figsize=(16, 9),
plt.title("Objevené exoplanety podle metody detekce a roku objevu")
plt.xlabel("Rok objevu")
plt.ylabel("Počet exoplanet")
plt.xticks(rotation=45)
plt.legend(title='Metoda detekce', bbox_to_anchor=(1.01, 1), loc='upper left')
plt.tight_layout()
plt.savefig("graphs/bar_plot_detection_method_by_year.png")
plt.close()
```

Select a načtení probíhá stejně jako u předchozího grafu, akorát na jinou dimenzionální tabulku.



Obrázek 16: Sloupkový graf typů metod detekcí exoplanet podle roku

Z grafu je vidět, že předtím se většinou exoplanety objevovaly se pomocí metody radial velocity (dopplerovská spektroskopie), poté se poprvé od roku 2002 začala používat tranzitní metoda. Do roku 2009 se planety převážně objevovaly pomocí dopplerovské spektroskopie. Od roku 2010 začala převažovat tranzitní metoda. Občas se i použila metoda direct imaging nebo od 2016 se i občas použila metoda gravitational microlensing. V roce 2023 i začla nová metoda *disk kinematics*. Podle tohoto grafu jsem došel k závěru, že typ

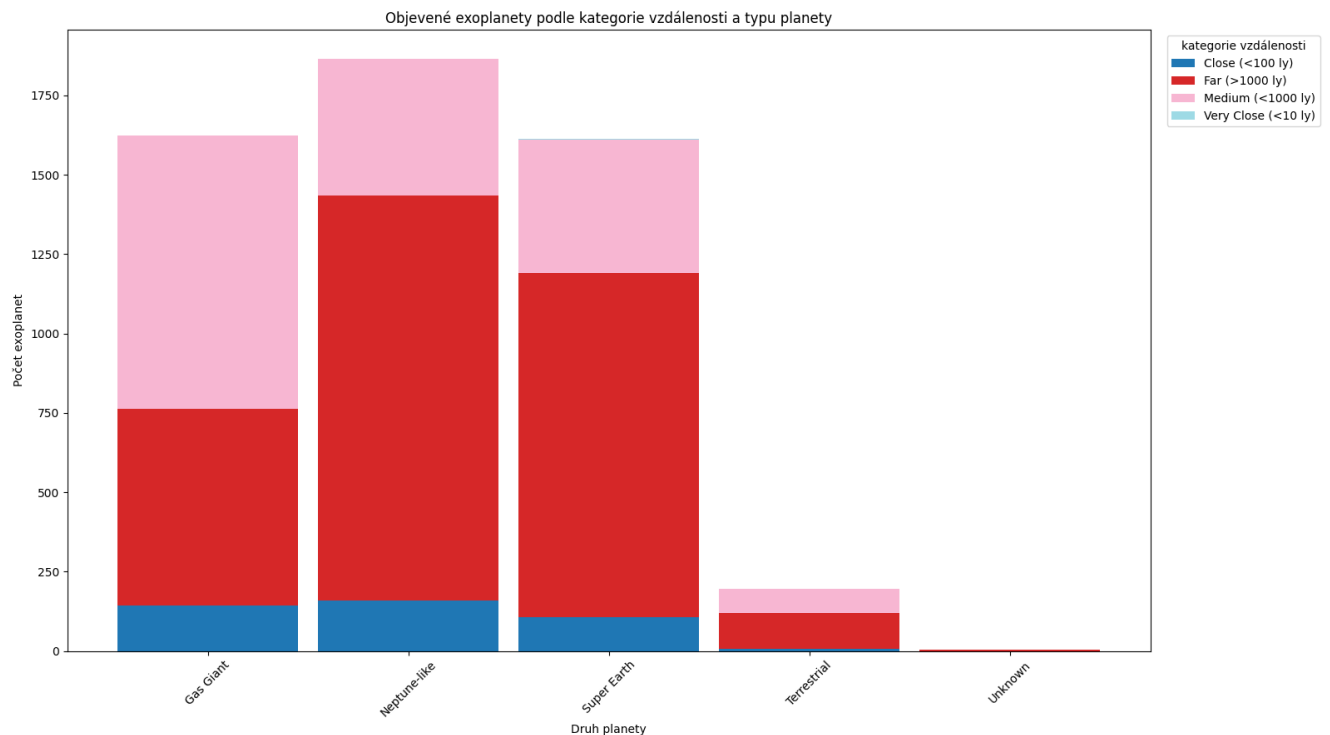
objevené planety souvisí s technologií, jaká se právě v tech letech používala.

11.5 Sloupcový graf počtu objevených exoplanet podle množství jejich typu s kategoriemi jejich vzdálenosti

```
con.execute("""
COPY_(
SELECT
dd.distance_category,
e.planet_type,
COUNT(*) AS num_planets
FROM exoplanets_e
JOIN dim_distance_category_dd ON e.distance_category_id = dd.distance_category_id
GROUP BY e.planet_type, dd.distance_category
ORDER BY e.planet_type
) TO 'results/exoplanet_distance_cat_type_count.parquet' (FORMAT 'parquet')
""")

df = con.execute("""SELECT * FROM 'results/exoplanet_distance_cat_type_count.parquet'""")
#print(df)

df_pivot = df.pivot(index='planet_type', columns='distance_category', value='num_planets')
df_pivot.plot(kind='bar', stacked=True, colormap='tab20', figsize=(16, 9),
plt.title("Objevené exoplanety podle kategorie vzdálenosti a typu planety")
plt.xlabel("Druh planety")
plt.ylabel("Počet exoplanet")
plt.xticks(rotation=45)
plt.legend(title='kategorie vzdálenosti', bbox_to_anchor=(1.01, 1), loc='upper right')
plt.tight_layout()
plt.savefig("graphs/bar_plot_planet_type_distance.png")
plt.close()
```



Obrázek 17: Sloupcový graf typů planet rozdělený do kategorií vzdálenosti

V tomto grafu jsme téměř žádnou souvislost nenašli. Jde vidět, že stejný druh planety se vyskytuje i daleko ať už je to planeta typu Neptunu, plynný obr nebo Superzemě. Najdeme téměř stejné množství i do vzdálenosti 100 ly, 1000 ly nebo ve vzdálenosti víc, jak 1000 ly. Z tohoto grafu můžu pouze říci, že skoro žádná planeta se nenachází blízko naší země do 10 ly.

11.6 Sloupcový graf objevených exoplanet podle typu a délky orbitální periody

```
con.execute("""
COPY_(
SELECT
dp.planet_type ,
oc.period_class ,
COUNT(*) AS_num_planets
FROM exoplanets_e
JOIN_dim_planet_type_dp_ON_e.planet_type_id=dp.planet_type_id
JOIN_dim_orbit_category_oc_ON_e.orbit_category_id=oc.orbit_category_id
GROUP_BY dp.planet_type , oc.period_class
ORDER_BY dp.planet_type
) TO 'results/exoplanet_planet_type_orbit_period_count.parquet' (FORMAT
""")
```

```

df = con.execute("""
SELECT_*_FROM_ 'results/exoplanet_planet_type_orbit_period_count.parquet
""").df()

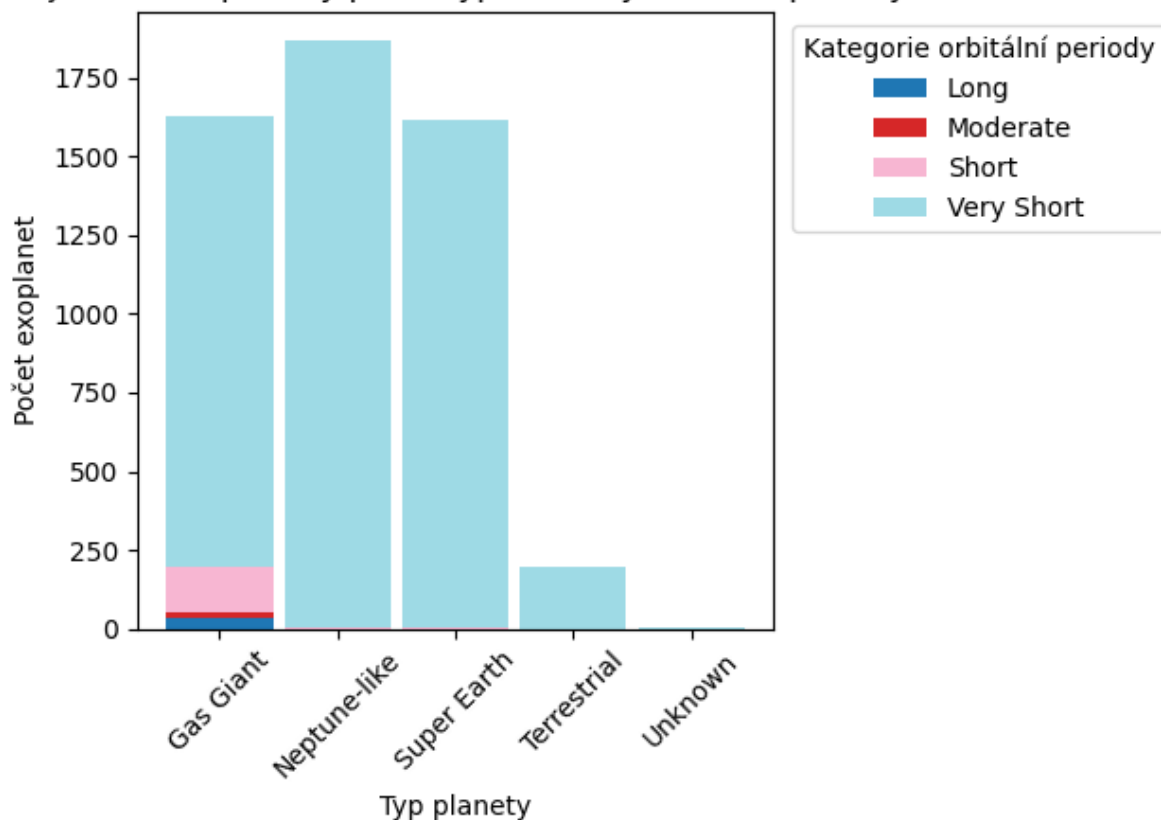
df_pivot = df.pivot(index='planet_type', columns='period_class', values='nu

plt.figure(figsize=(16, 9))
df_pivot.plot(kind='bar', stacked=True, colormap='tab20', width=0.9)

plt.title("Objevene_exoplanety_podle_typu_a_delky_orbitalni_periody")
plt.xlabel("Typ_planety")
plt.ylabel("Pocet_exoplanet")
plt.xticks(rotation=45)
plt.legend(title='Kategorie_orbitalni_periody', bbox_to_anchor=(1.01, 1), 1
plt.tight_layout()
plt.savefig("graphs/bar_plot_planet_type_orbit_period.png")
plt.close()

```

Objevené exoplanety podle typu a délky orbitální periody



Obrázek 18: Sloupcový graf objevených exoplanet podle typu a délky orbitální periody

V tomto grafu je vidět, že se pouze občas plynní obři mají dlouhou periodu objevu,

kratší nebo střední, jinak ostatní typy Superzemě, Terestriální a typu Neptun mají hodně krátkou periodu objevu. Zřídka může mít krátkou periodu planeta typu Neptunu a nebo Superzemě.