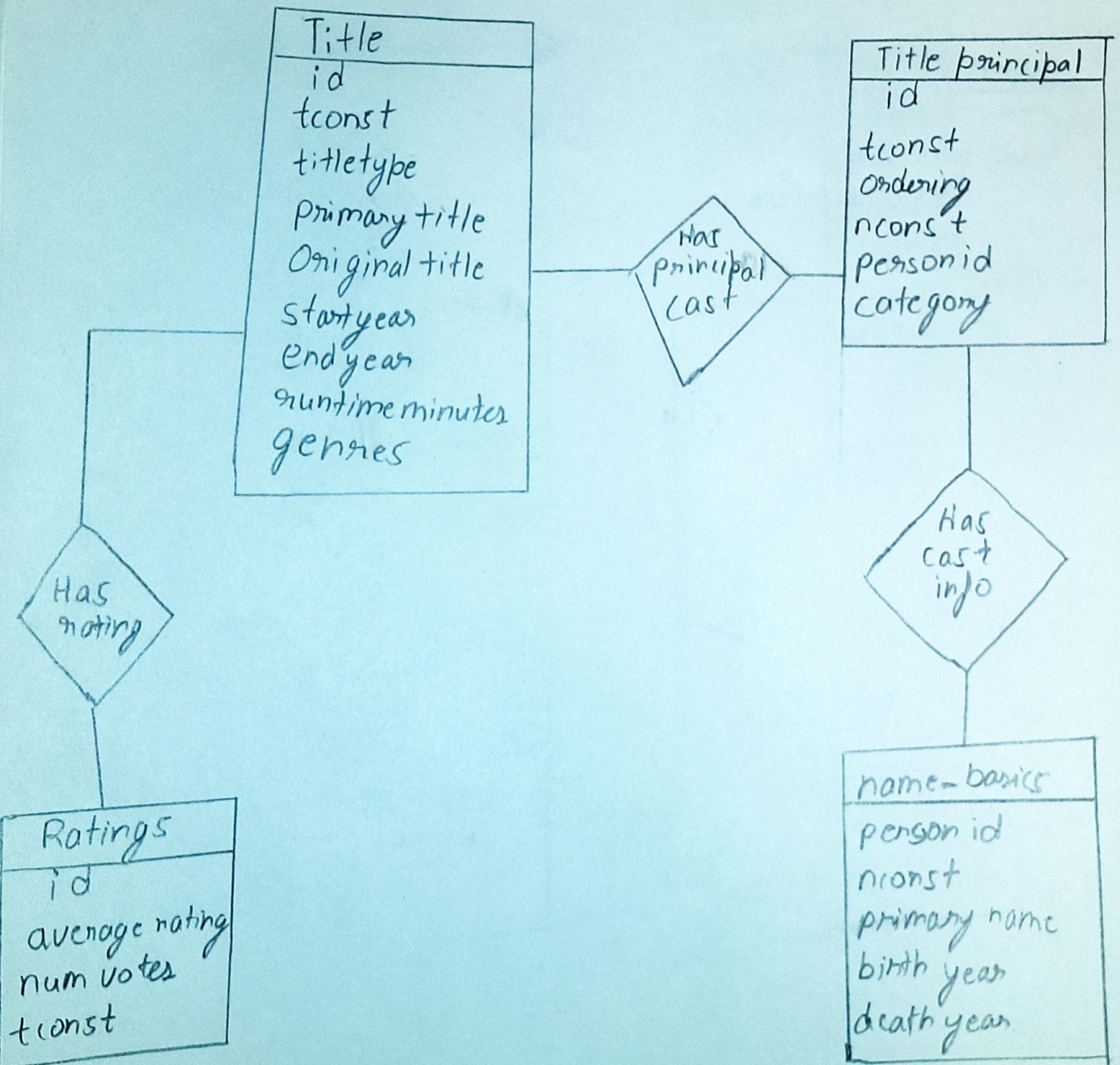


**CSCI 620/Section 03/Mior, Introduction to Big
Data, Spring 2215**

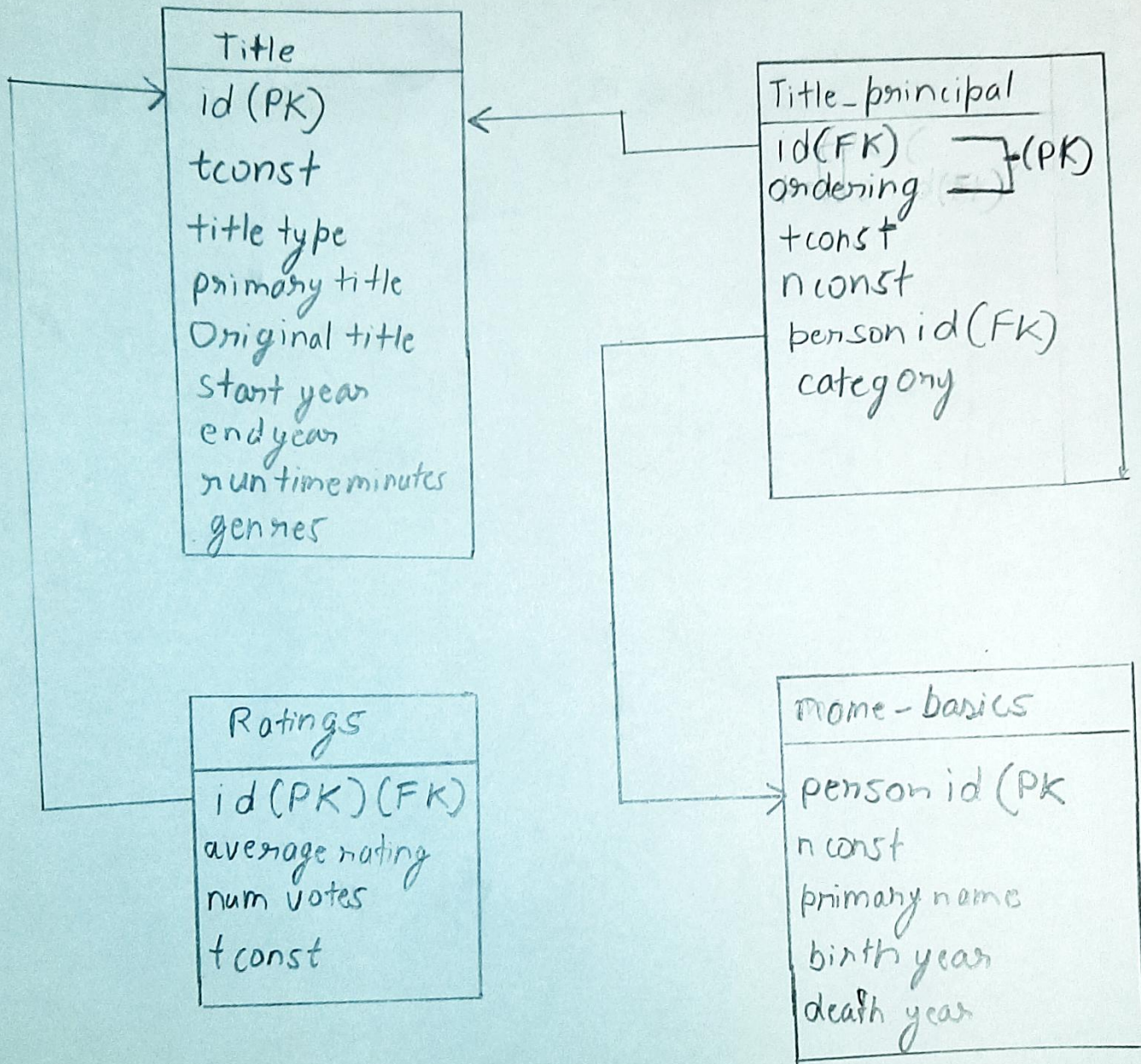
Assignment 1 – Relational model

Submitted by – Prakhar Gupta(pg9349)

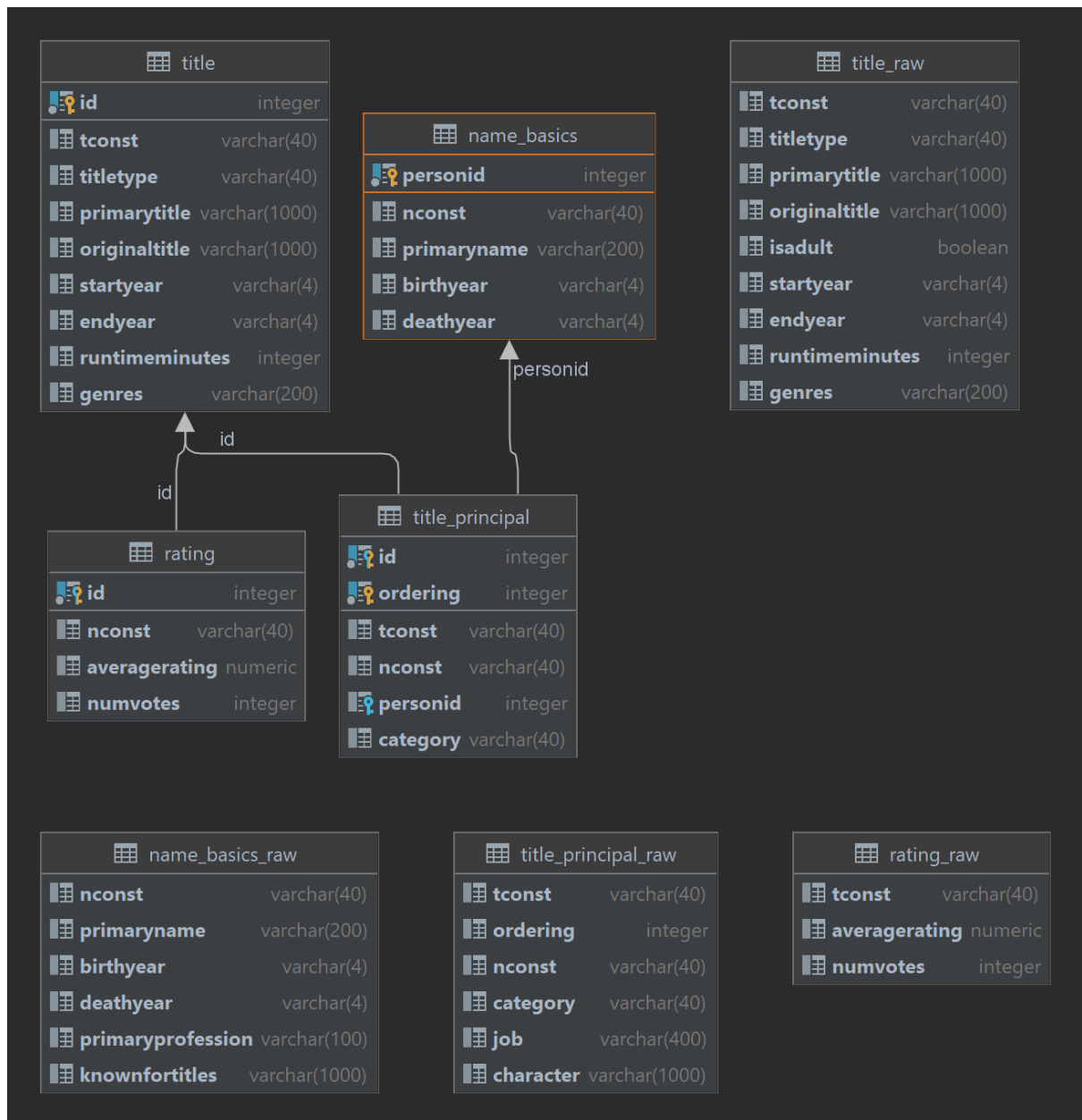
Am1) ER Diagram



Ans 2) Relationship Model (Final Tables)



Detailed relationship model (Contains all tables raw and final in the DB)



Refer to code 1 and code 3 provide in the folder-

To handle primary keys and to use integer as primary key we have sliced the ids which are tconst and nconst using substr function to get numeric part then we have used cast function to cast them into integer

Ans3

IMDB data base consist of 7 tsv file which are imported in the database. Additional analysis is done beside general details

Main Files use-

The columns in italics are not present in final tables

Columns in blue are added in database which are not present in file

title.basics.tsv – Information of the titles like runtime and genre .we have filter the data set to only contain movies which are non adult:

- tconst (string) - alphanumeric unique identifier of the title
- **id**(number) –sliced primary key created using tconst
- titleType (string) – Type of title (Filter condition was movie)
- primaryTitle (string) – title used for marketing
- originalTitle (string) - original title, in the original language
- *isAdult (boolean)* - 0: non-adult title; 1: adult title (Filter condition was 0)
- startYear (YYYY) – Release year
- endYear (YYYY) – End year
- runtimeMinutes – Run time in minutes
- genres (string array) – includes up to 3 genres that describe the movie

title.principals.tsv – Contains the principal cast/crew for titles. This file acts as a map for title mapping with their cast

- tconst (string) - alphanumeric unique identifier of the title
- **id**(number) –sliced primary key created using tconst. It also is a foreign key which refers to title basics. Also part of composite key
- ordering (integer) – id to represent each member of a cast for each tconst part of composite key
- nconst (string) - alphanumeric unique identifier of the name/person
- **personid**(number) – Foreign key from name_basics table
- category (string) – The job of the cast member in broad sense(Filter condition was 'director','producer','writer','actor','actress') This column tell us what role did that person play in creating of a particular movie
- *job (string)* - the specific job of a person
- *characters (string)* - the name of the character played by that cast

title.ratings.tsv –IMDb rating and votes for titles. This contains average ratings and number of votes

- tconst (string) - alphanumeric unique identifier of the title

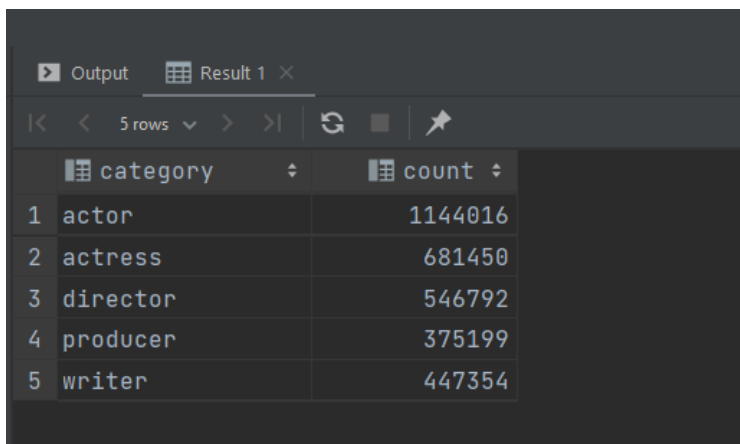
- `id(number)` – sliced primary key created using `tconst`. It also is a foreign key which refers to `title basics`
- `averageRating` – a weighted average of all the individual user ratings
- `numVotes` – number of votes the title has received

name.basics.tsv – This file contains the details for the cast involved like name birth year and death year if present.

- `nconst (string)` – alphanumeric unique identifier of the name/person
- `personid(number)` – sliced primary key created using `nconst`
- `primaryName (string)` – name of the person
- `birthYear` – in YYYY format
- `deathYear` – in YYYY format if applicable, else `'\N'`
- `primaryProfession (array of strings)` – the top-3 professions of the person
- `knownForTitles (array of tconsts)` – titles the person is known for

Some data exploration of the above tables(refer to `analysis.sql` file for code) –

1. Individual cast counts –



The screenshot shows a database query result with two columns: 'category' and 'count'. The results are as follows:

	category	count
1	actor	1144016
2	actress	681450
3	director	546792
4	producer	375199
5	writer	447354

2. Top 10 Movies with most `main('director','producer','writer','actor','actress')` cast

	id	primarytitle	"No of main cast"
1	1115	Ansigttyven I	10
2	1116	Ansigttyven II	10
3	1175	Camille	10
4	1240	Hamlet	10
5	1592	In the Prime of Life	10
6	2026	Anny - Story of a Prostitute	10
7	2101	Cleopatra	10
8	2153	The Great Circus Catastrophe	10
9	2211	Den glade løjtnant	10
10	2405	Oliver Twist	10

3. Runtimes of the movies(min max and average)

	min	max	avg
1	1	51420	90.023718774628604

4. Min max and average no of votes in ratings table

	"min no of votes"	"Max no of Votes"	"Average no of votes"
1	5	2537419	3518.0494401914052807

These file were not use as they either had redundant information or was not required-

title.akas.tsv- Not required

title.crew.tsv – Contains detail info of the crew involved we using the category column from title principal table

title.episode.tsv- Contains the tv episode information. We have limited the scope to non adult movies

Ans4-

Refer to codes3 and code 4 in the folder.

To solve the problem of foreign key violations we are filtering the title principal raw table for ids which were present in title table and person id which were present in name table and only inserted those in your final table which matched.

Regarding the speed which we have created a data pipe line which as a staging area to load raw table we can run our code end to end in approximately 17 minutes.

Timings for each code are provided below-

- **code1_create_raw_tables.sql** – This code create raw table which act as base for our future data steps
Runtime -109 ms
- **Code2_import_into_raw.sql** – Using postgresql copy command we copy table from the tsv files into tables created in code1. Relevant uncompressing and permissions should be granted before hand
Runtime - 2 min, 5 sec, 27 ms
- **Code3_create_final_tables.sql**- This code create final tables they will be later used as placeholder for data insertions
Runtime - 62 ms
- **Code4_import_into_final.sql**- This code inserts values into final tables after relevant preprocessing
Runtime - 14 min, 19 sec, 960 ms

Ans5

Refer to code in folder q5.py