# Indices

item_price_index in table Item (btree): on column item_price
This was created as I expect a common search query will be price, either wanting to see prices in ascending or descending order, or to see a certain range of prices for each item. I chose a btree for this index as it is best for both range queries and ordered sort.

employee_role_index in Table employee (hash) on column job_category
This was created as I expect a common search query will be what jobs each employee has at the company. This would be useful because what if you are making a new team and need to assign people to said team, a query looking at certain jobs and who works them would be quite useful. I chose hash index as it would be best for looking up certain types of jobs.

## Additional indices due to UNIQUE

This part mentions indices that were created due to the unique constraint on certain columns during their creation due to them being candidate keys. I'm not sure if they would qualify being here but I'm adding a section for them just in case, especially since I would have added employee_email_key and user_email_key deliberately if unique didn't do that for me.

company_url_key in table company (btree):
This was would be useful as finding the url of each company would be useful in getting to its website, I would also expect this to frequent query. Btree is used here due to the amount of unique urls that would exist here, as well as being better for ordering companies by url

employee_email_key in table employee (btree):
This is useful as you can sort employees by email, making it easer to find them, it would also most likely be rather frequent query. Btree is used here due to amount of unique emails being in this column.

users_user_email_key in table users (btree):
This is useful as you can sort users by email, making it easer to find them, it would also most likely be rather frequent query . Btree is used here due to amount of unique emails being in this column.