1) List all employees, i.e. all tuples in the jbemployee relation

Selecting all the employees name from name column.

**Query:**

**SELECT name FROM jbemployee;**

Graphical user interface, text, application

Description automatically generated

2)List the name of all departments in alphabetical order. Note: by “name” we mean the name attribute for all tuples in the jbdept relation.

The names are listed in ascending order.

**Query:**

**SELECT name FROM jbdept ORDER BY name;**

Graphical user interface, application

Description automatically generated

3)What parts are not in store, i.e. qoh = 0? (qoh = Quantity On Hand)

Listing any parts from the jbparts table that do not have a qoh in store.

**Query:**

**SELECT \* FROM jbparts WHERE qoh=0;**

Graphical user interface, text, application

Description automatically generated

4)Which employees have a salary between 9000 (included) and 10000 (included)?

Selecting the employee name whose salary between 9000 & 10000.

**Query:**

**SELECT id,name,salary**

**FROM jbemployee**

**WHERE salary between 9000 AND 10000;**

Graphical user interface, text, application

Description automatically generated

5)What was the age of each employee when they started working (startyear)? Finding the age of an employee when they started working by using 2 columns (startyear, and birthyear)

Finding the age of the employee by using started year of the company.

**Query:**

**SELECT Name, startyear - Birthyear AS Employee\_experience FROM jbemployee;**

Graphical user interface

Description automatically generated with medium confidence

6) Which employees have a last name ending with “son”?

Finding the employee whose last name ending with “son”.

**Query:**

**SELECT name**

**FROM jbemployee**

**WHERE name LIKE ‘%son, %’;**

Graphical user interface, text, application, email

Description automatically generated

7) Which items (note items, not parts) have been delivered by a supplier called Fisher-Price? Formulate this query using a subquery in the where-clause.

Using subquery to get the items have been delivered by the supplier.

**Query:**

**SELECT id,name,supplier FROM jbitem**

**WHERE supplier in (SELECT id FROM jbsupplier WHERE name='fisher-price')**

**ORDER BY name ASC;**

Graphical user interface, text, application, email

Description automatically generated

8)Formulate the same query as above, but without a subquery.

Inner join is used here to get the supplier.

**Query:**

**SELECT jbitem.id, jbitem.name**

**from jbitem inner join jbsupplier on jbitem.supplier=jbsupplier.id**

**WHERE jbsupplier.name='fisher-price';**

Graphical user interface, text, application

Description automatically generated

9) Show all cities that have suppliers located in them. Formulate this query using a subquery in the where-clause.

Selecting cities using subquery where the id of a city is in supplier table.

**Query:**

**SELECT \* FROM jbcity**

**where id in (SELECT city FROM jbsupplier);**

Graphical user interface, application

Description automatically generated

10) What is the name and color of the parts that are heavier than a card reader?

In order to obtain the name and color of the specific parts, a subquery is utilized to retrieve their weight.

**Query:**

**SELECT name, color**

**FROM jbparts**

**WHERE weight> (SELECT weight FROM jbparts WHERE name='card reader');**

Graphical user interface, text, application

Description automatically generated

11) Formulate the same query as above, but without a subquery. (The query must not contain the weight as a constant.)

Obtain the name and color of the specific parts without the subquery

**Query:**

**SELECT tab1.name, tab1.color**

**FROM jbparts tab1 JOIN jbparts tab2**

**WHERE tab2.name='card reader' AND tab1.weight>tab2.weight;**

Graphical user interface, text, application

Description automatically generated

12) What is the average weight of black parts?

The average is obtained by utilizing the avg function and applying a where-clause for a particular color.

**Query:**

**SELECT avg(weight) AS Average\_weight FROM jbparts**

**WHERE color='black';**

Graphical user interface, text, application

Description automatically generated

13) What is the total weight of all parts that each supplier in Massachusetts (“Mass”) has delivered? Retrieve the name and the total weight for each of these suppliers. Do not forget to take the quantity of delivered parts into account. Note that one row should be returned for each supplier.

Using inner join to merge the tables to find the total weight of each supplier in Massachusetts (mass).

**Query:**

**SELECT jbsupplier.name as Supplier\_name, SUM(jbsupply.quan \* jbparts.weight) as TotalWeight**

**FROM jbsupplier**

**INNER JOIN jbcity ON jbsupplier.city = jbcity.id**

**INNER JOIN jbsupply ON jbsupplier.id = jbsupply.supplier**

**INNER JOIN jbparts ON jbsupply.part = jbparts.id**

**WHERE jbcity.state = 'Mass'**

**GROUP BY jbsupplier.id**

**ORDER BY Supplier\_name ASC;**

Graphical user interface, text, application, email

Description automatically generated

14) Create a new relation (a table), with the same attributes as the table items using the CREATE TABLE syntax where you define every attribute explicitly (i.e. not as a copy of another table). Then fill the table with all items that cost less than the average price for items. Remember to define primary and foreign keys in your table!

Creating the table with the same attribute as jbitem and added the values in the table where the cost is less than the average price by defining the primary and foreign key.

**Query:**

**CREATE TABLE jbitems\_cost (**

**id int,**

**Name varchar(255),**

**Dept int not null,**

**Price int not null,**

**qoh int,**

**Supplier int not null,**

**primary key(id), foreign key(dept) references jbdept(id),**

**foreign key(supplier) references jbsupplier(id)**

**);**

**INSERT into jbitems\_cost (id, name, dept, price, qoh, supplier)**

**SELECT id, name, dept, price, qoh, supplier from jbitem**

**WHERE price < (SELECT AVG(price) FROM jbitem);**

Graphical user interface, text, application

Description automatically generated