

CheckPoint Queries

Queries from Checkpoint 2

- Find the titles of all books by Pratchett that cost less than \$10.

```
SELECT B.b_id, B.title
FROM BOOK AS B
WHERE (SELECT A.b_id
      FROM AUTHOR AS A, NAMES AS N
      WHERE A.name_id == N.id AND
            N.lname LIKE 'Pratchett%' AND
            A.b_id == B.b_id) == B.b_id AND
      B.price <= 10;
```

- Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer).

```
SELECT OIT.bid, B.title, OIT.timestamp, OIT.quantity
FROM BOOK AS B, ORDER_TRANSACTION AS O, ORDERITEM AS OIT
WHERE
  (SELECT C.entity_id
   FROM CUSTOMER AS C, ENTITY AS E, NAMES AS N
   WHERE E.name_id == N.id AND
         C.entity_id == E.id AND
         N.fname == 'Jane' AND
         N.lname == 'Gonzalez') == O.customer_id AND
  O.order_id == OIT.order_id AND
  OIT.bid == B.b_id;
```

- Find the titles and ISBNs for all books with less than 5 copies in stock.

```
SELECT ISBN, title
FROM ISBN AS I, BOOK AS B, INVENTORY AS IV
WHERE IV.quantity < 5 AND
      IV.bid = I.b_id AND
      I.b_id = B.b_id;
```

- Give all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased.

```
SELECT NAMES.fname, NAMES.lname, B.title
FROM BOOK AS B, NAMES
WHERE B.b_id IN (SELECT I.b_id
                FROM AUTHOR AS A, NAMES, ISBN AS I
                WHERE A.name_id = NAMES.id AND
                      NAMES.lname LIKE '%Pratchett%' AND
                      I.b_id = A.b_id) AND
  (SELECT E.name_id
   FROM ENTITY AS E, ORDERITEM AS O, ORDER_TRANSACTION AS T, CUSTOMER AS C
   WHERE B.b_id = O.bid AND
         O.order_id = T.order_id AND
         T.customer_id = C.entity_id AND
```

```
C.entity_id = E.id) = NAMES.id;
```

- Find the total number of books purchased by a single customer (you choose how to designate the customer).

```
SELECT B.fname, B.lname, SUM(quantity) AS Total_Quantity
FROM ORDERITEM AS O, (SELECT order_id, D.fname, D.lname
                      FROM ORDER_TRANSACTION AS T, (SELECT C.entity_id, N.fname, N.lname
                                                    FROM CUSTOMER AS C, ENTITY AS E, NAMES AS N
                                                    WHERE N.fname LIKE '%Jane' AND
                                                          N.lname LIKE '%Gonzalez' AND
                                                          N.id = E.name_id AND
                                                          E.id = C.entity_id) AS D
                      WHERE T.customer_id = D.entity_id ) AS B
WHERE O.order_id = B.order_id
```

```
SELECT NAMES.fname, NAMES.lname, SUM(quantity) AS Total_Quantity
FROM ORDERITEM AS O, NAMES
WHERE O.order_id IN (SELECT order_id
                    FROM ORDER_TRANSACTION AS T
                    WHERE T.customer_id IN (SELECT C.entity_id
                                            FROM CUSTOMER AS C, ENTITY AS E, NAMES AS N
                                            WHERE N.fname LIKE '%Jane' AND
                                                  N.lname LIKE '%Gonzalez' AND
                                                  N.id = E.name_id AND
                                                  E.id = C.entity_id)) AND

NAMES.fname LIKE '%Jane' AND
NAMES.lname LIKE '%Gonzalez';
```

- Find the customer who has purchased the most books and the total number of books they have purchased.

```
SELECT fname, lname, C.max
FROM (SELECT customer_id, max(B.customer_total) AS max
      FROM (SELECT customer_id, sum(Total) AS customer_total
            FROM (SELECT T.order_id, customer_id, Total
                  FROM (SELECT order_id, SUM(quantity) AS Total
                        FROM ORDERITEM AS O
                        GROUP BY O.order_id) AS A JOIN ORDER_TRANSACTION AS T
                  ON A.order_id = T.order_id)
            GROUP BY customer_id) AS B) AS C, CUSTOMER, ENTITY AS E, NAMES AS N
WHERE C.customer_id = CUSTOMER.entity_id AND
      CUSTOMER.entity_id = E.id AND
      E.name_id = N.id;
```

- Three more Additional Queries (include joins and at least one include aggregate function and at least one Queries use 'extra' Entities from Checkpoint 1).

Queries from Checkpoint 3

- Provide a list of customer names, along with the total dollar amount each customer has spent.

```
SELECT N.id, fname, lname, C.c_total AS total_spend
FROM (SELECT customer_id, B.customer_total AS c_total
      FROM (SELECT customer_id, sum(Total) AS customer_total
            FROM (SELECT T.order_id, customer_id, Total
```

```

FROM (SELECT order_id, SUM(quantity * B.price) AS Total
      FROM ORDERITEM AS O, BOOK AS B
      WHERE B.b_id = O.bid
      GROUP BY O.order_id) AS A JOIN ORDER_TRANSACTION AS T
      ON A.order_id = T.order_id
GROUP BY customer_id) AS B) AS C, CUSTOMER, ENTITY AS E, NAMES AS N
WHERE C.customer_id = CUSTOMER.entity_id AND
      CUSTOMER.entity_id = E.id AND
      E.name_id = N.id;

```

- Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.

```

WITH CustTotals AS (SELECT customer_id, ROUND(sum(Total), 2) AS customer_total
                    FROM (SELECT T.order_id, customer_id, Total
                          FROM (SELECT order_id, SUM(quantity * B.price) AS Total
                                FROM ORDERITEM AS O, BOOK AS B
                                WHERE B.b_id = O.bid
                                GROUP BY O.order_id) AS A JOIN ORDER_TRANSACTION AS T
                                ON A.order_id = T.order_id)
                          GROUP BY customer_id
                    ORDER BY customer_total DESC)

SELECT *
FROM CustTotals
WHERE customer_total > (SELECT AVG(customer_total) FROM (CustTotals))
ORDER BY customer_total DESC

```

- Provide a list of the titles in the database and associated total copies sold to customers, sorted from the title that has sold the most individual copies to the title that has sold the least.

```

SELECT title, C.total_quantity
FROM BOOK AS B, (SELECT bid, SUM(quantity) AS total_quantity
                  FROM ORDERITEM AS O
                  GROUP BY O.bid) AS C
WHERE B.b_id = C.bid
ORDER BY C.total_quantity DESC

```

- Provide a list of the titles in the database and associated dollar totals for copies sold to customers, sorted from the title that has sold the highest dollar amount to the title that has sold the smallest.

```

SELECT title, (C.total_quantity * B.price) AS total_dollar_soldn
FROM BOOK AS B, (SELECT bid, SUM(quantity) AS total_quantity
                  FROM ORDERITEM AS O
                  GROUP BY O.bid) AS C
WHERE B.b_id = C.bid
ORDER BY C.total_quantity DES

```

- Find the most popular author in the database (i.e. the one who has sold the most books).

```

SELECT fname, lname, BD.title
FROM (SELECT fname, lname, A.b_id
      FROM (AUTHOR AS A JOIN NAMES AS N ON (A.name_id = N.id))
      GROUP BY A.name_id) AS NA, (SELECT B.b_id AS bid, max(C.total_quantity), B.title
                                FROM BOOK AS B, (SELECT bid, SUM(quantity) AS total_quantity
                                                  FROM ORDERITEM AS O

```

```

                                GROUP BY O.bid) AS C
                                WHERE B.b_id = C.bid) AS BD
WHERE NA.b_id = BD.bid

```

- Find the most profitable author in the database for this store (i.e. the one who has brought in the most money).

```

SELECT fname, lname, BD.title
FROM (SELECT fname, lname, A.b_id
      FROM (AUTHOR AS A JOIN NAMES AS N ON (A.name_id = N.id))
      GROUP BY A.name_id) AS NA, (SELECT B.b_id AS bid, max(C.total_quantity * B.price),
      B.title
                                FROM BOOK AS B, (SELECT bid, SUM(quantity) AS total_quantity
                                FROM ORDERITEM AS O
                                GROUP BY O.bid) AS C
                                WHERE B.b_id = C.bid) AS BD
WHERE NA.b_id = BD.bid

```

- Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.

```

SELECT E.id, NAMES.fname, NAMES.lname, E.address, E.city, E.state, E.country, E.email, E.phone,
E.postalcode
FROM ENTITY AS E, (SELECT DISTINCT T.customer_id
                   FROM (SELECT DISTINCT O.order_id
                         FROM ORDERITEM AS O
                         WHERE O.bid IN (SELECT DISTINCT A.b_id
                                         FROM AUTHOR AS A
                                         WHERE A.name_id IN (SELECT DISTINCT NA.name_id
                                                             FROM (SELECT A.name_id, A.b_id
                                                                    FROM (AUTHOR AS A JOIN
                                                                    GROUP BY A.name_id) AS NA,
                                                                    (SELECT B.b_id AS bid, max(C.total_quantity * B.price), B.title
                                                                    FROM BOOK AS B, (SELECT bid, SUM(quantity) AS total_quantity
                                                                    FROM ORDERITEM AS O
                                                                    GROUP BY O.bid) AS C
                                                                    WHERE B.b_id = C.bid) AS BD
                                                                    WHERE NA.b_id = BD.bid)))) AS OID
                   JOIN ORDER_TRANSACTION AS T ON (OID.order_id = T.order_id)) AS TID, NAMES
WHERE TID.customer_id = E.id AND
      NAMES.id = E.name_id

```

- Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```

WITH CustTotals AS (SELECT customer_id, ROUND(sum(Total), 2) AS customer_total
                   FROM (SELECT T.order_id, customer_id, Total
                         FROM (SELECT order_id, SUM(quantity * B.price) AS Total
                               FROM ORDERITEM AS O, BOOK AS B
                               WHERE B.b_id = O.bid
                               GROUP BY O.order_id) AS A JOIN ORDER_TRANSACTION AS T

```

```
                ON A.order_id = T.order_id)
            GROUP BY customer_id
            ORDER BY customer_total DESC)
SELECT DISTINCT NAMES.fname AS First_name, NAMES.lname AS Last_name
FROM CustTotals AS C, (SELECT ORDER_TRANSACTION.order_id FROM ORDER_TRANSACTION) AS OT,
ORDERITEM AS OIT, AUTHOR AS AU, NAMES
WHERE C.customer_id = OT.order_id AND
      OT.order_id = OIT.order_id AND
      AU.b_id = OIT.bid AND
      AU.name_id = NAMES.id;
```

Checkpoint VIEWS
