## Aufgabe 1

Α

SELECT FILIALE, EXTRACT(month FROM DATUM) AS Monat, SUM(PREIS) AS Umsatz FROM BONDATEN GROUP BY FILIALE, EXTRACT(month FROM DATUM);

	FILIALE	MONAT	UMSATZ
1	Α	10	22,96
2	Α	9	534,53
3	В	9	2655,68
4	В	10	9,96

В

SELECT FILIALE, ARTIKEL.NAME, COUNT(ARTIKEL) AS Menge FROM BONDATEN, ARTIKEL WHERE BONDATEN.ARTIKEL = ARTIKEL.ID AND ARTIKEL.NAME = 'Handcreme' AND DATUM BETWEEN TO\_DATE('01.09.2012', 'dd.mm.yyyy') AND TO\_DATE('03.09.2012', 'dd.mm.yyyy') GROUP BY FILIALE, ARTIKEL.NAME ORDER BY Menge DESC;

	FILIALE	NAME	MENGE
1	Α	Handcreme	14
2	В	Handcreme	8

С

SELECT ARTIKEL.ARTIKELGRUPPE, SUM(BONDATEN.PREIS) AS Umsatz FROM ARTIKEL, BONDATEN WHERE BONDATEN.ARTIKEL = ARTIKEL.ID GROUP BY ARTIKEL.ARTIKELGRUPPE;

	ARTIKELGRUPPE	UMSATZ
1	К	563,34
2	S	2531,71
3	G	128,08

## Aufgabe 2

```
CREATE TABLE PREISE (
  ID NUMBER(10) NOT NULL,
  PREIS NUMBER(10,2),
  DATUM DATE,
  CONSTRAINT PREISE PK PRIMARY KEY(ID)
);
ALTER TABLE ARTIKEL
ADD (PREIS number(10),
    CONSTRAINT ARTIKEL_PREIS FOREIGN KEY (PREIS) REFERENCES
PREISE(ID)
    );
CREATE TRIGGER BONDATEN_PREIS
  BEFORE INSERT OR UPDATE ON BONDATEN
  FOR EACH ROW
  DECLARE
    p_preis number(10,2);
  BEGIN
    if(:new.preis is null) then
      select PREISE.PREIS INTO p_preis
      FROM PREISE, ARTIKEL
      WHERE :new.ARTIKEL = ARTIKEL.ID
      AND ARTIKEL.PREIS = PREISE.ID;
      :new.preis := p_preis;
    END IF;
END;
/
```

## Aufgabe 3

```
ALTER TABLE ARTIKEL
ADD (BESTAND NUMBER(10));

CREATE TABLE LOGS (
  id number(10) not null,
  zeit TIMESTAMP(6) not null,
  text VARCHAR2(250) not null,
  CONSTRAINT LOGS_PL PRIMARY KEY (id)
);

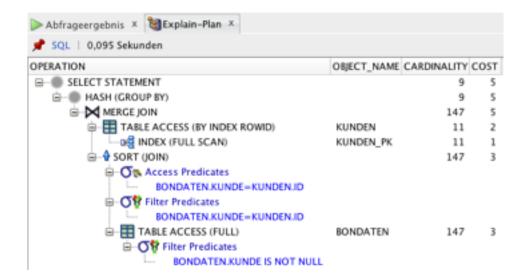
CREATE SEQUENCE LOGS_PK
  MINVALUE 0
  MAXVALUE 100000000
  START WITH 0
  INCREMENT BY 1
```

CACHE 20;

```
CREATE TRIGGER UPDATE_BESTAND
BEFORE INSERT OR UPDATE ON BONDATEN
FOR EACH ROW
  DECLARE
  artikel bestand number(10);
  bestell_text VARCHAR2(250);
  BEGIN
    SELECT BESTAND INTO artikel_bestand
    FROM ARTIKEL
    WHERE :new.ARTIKEL = ARTIKEL.ID;
    artikel bestand := artikel bestand - 1;
    UPDATE ARTIKEL
    SET BESTAND = artikel_bestand
    WHERE ARTIKEL.ID = :new.ARTIKEL;
    bestell_text := 'Bestellung für Artikel: ' + :new.ARTIKEL + '
Menge: 50';
    if(artikel_bestand <= 10) THEN</pre>
      UTL_MAIL.send(sender => 'florian.bauer@haw-hamburg.de',
            recipients => 'florian.bauer@posteo.de',
               subject => 'WP-DD Meldebestand ist erreicht',
               message => bestell_text,
             mime_type => 'text; charset=us-ascii');
      INSERT INTO LOGS
      (id,zeit,text) values (LOGS PK.NEXTVAL, SYSDATE,
bestell text);
    END IF;
END;
```

## Aufgabe 4

SELECT KUNDEN.ID, KUNDEN.VORNAME, KUNDEN.NACHNAME, SUM(BONDATEN.PREIS) AS GESAMTWERT FROM KUNDEN, BONDATEN WHERE BONDATEN.KUNDE = KUNDEN.ID GROUP BY KUNDEN.ID, KUNDEN.VORNAME, KUNDEN.NACHNAME;



Die Darstellung Explain-Plan zeigt die Arbeitsschritte als Baumstruktur auf. Es beginnt mit dem Select Statement als Root-Objekt gefolgt vom Group by. Unterhalb des Merge Teilt sich der Baum in zwei Zweige, wobei der erste, deutlich einfachere Zweig, den Zugriff auf die Kunden-Tabelle abbildet. Es wird hierbei keine weitere Filterung angewandt. Im zweiten Zweig wird auf die Bondaten-Tarbell zugegriffen mit dem Filter das BONDATEN.KUNDE nicht null sein darf. Die so erhaltene Auswahl wird weiter gefiltert nach BONDATEN.KUNDE = KUNDEN.ID und wird basierend auf diesem Filter mit der Kunden-Tabelle vermengt (MERGE).

Die Cost spalte gibt die Kosten an die eine Operation gekostet hat. Die Cardinality gibt die Anzahl der Zeilen an auf die zugegriffen wird.